



Dialogic® Converged Services Platform Release 8.4.1 Engineering Release 3

API Reference

Copyright and Legal Disclaimer

Copyright © [1998-2008] Dialogic Corporation. All Rights Reserved. You may not reproduce this document in whole or in part without permission in writing from Dialogic Corporation at the address provided below.

All contents of this document are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation or its subsidiaries. Reasonable effort is made to ensure the accuracy of the information contained in the document. However, due to ongoing product improvements and revisions, Dialogic Corporation and its subsidiaries do not warrant the accuracy of this information and cannot accept responsibility for errors or omissions that may be contained in this document.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH DIALOGIC® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS EXPLICITLY SET FORTH BELOW OR AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

It is possible that the use or implementation of any one of the concepts, applications, or ideas described in this document, in marketing collateral produced by or on web pages maintained by Dialogic Corporation or its subsidiaries may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic Corporation or its subsidiaries do not provide any intellectual property licenses with the sale of Dialogic products other than a license to use such product in accordance with intellectual property owned or validly licensed by Dialogic Corporation or its subsidiaries. More detailed information about such intellectual property is available from Dialogic Corporation's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. The software referred to in this document is provided under a Software License Agreement. Refer to the Software License Agreement for complete details governing the use of the software.

Dialogic Corporation encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.

Dialogic, Dialogic Pro, Brooktrout, Cantata, SnowShore, Eicon, Eicon Networks, Eiconcard, Diva, SIPcontrol, Diva ISDN, TruFax, Realblobs, Realcomm 100, NetAccess, Instant ISDN, TRXStream, Exnet, Exnet Connect, EXS, ExchangePlus VSE, Switchkit, N20, Powering The Service-Ready

Network, Vantage, Connecting People to Information, Connecting to Growth, Making Innovation Thrive, and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic.

Windows NT is a registered trademarks of Microsoft Corporation in the United States and/or other countries. Other names of actual companies and products mentioned herein are the trademarks of their respective owners.

This document discusses one or more open source products, systems and/or releases. Dialogic is not responsible for your decision to use open source in connection with Dialogic products (including without limitation those referred to herein), nor is Dialogic responsible for any present or future effects such usage might have, including without limitation effects on your products, your business, or your intellectual property rights.

Dialogic Product Line Warranty

Unless otherwise stated in an applicable product purchase agreement between the Customer and Dialogic, Dialogic warrants that during the Warranty Period, products will operate in substantial conformance with Dialogic's standard published documentation accompanying the product. If a product does not operate in accordance therewith during the Warranty Period, the Customer must promptly notify Dialogic. Dialogic, at its option, will either repair or replace the product without charge. The Customer has the right, as their exclusive remedy, to return the product for a refund of purchase price or license fee if Dialogic is unable to repair or replace it.

Warranty Period

In the event that you have no signed agreement setting out a warranty period, the Warranty Period shall be the standard warranty period set out on www.dialogic.com on the date of your purchase of the product.

The Warranty Period begins on the date of shipment of any products or software by Dialogic.

The Warranty Period for repaired, replaced or corrected products and software shall be coterminous to the Warranty Provided for the original products or software purchased.

To report warranty claims, Customer may contact Dialogic via email at techsupport@cantata.com or call (781) 433-9600.

Warranty Provisions

A. During the Warranty Period, Dialogic warrants to Customer only that:

- (i) Products manufactured by Dialogic (including those manufactured for Dialogic by an original equipment manufacturer) will be free from defects in material and workmanship and will substantially conform to specifications for such products;
- (ii) software developed by Dialogic will be free from defects which materially affect performance in accordance with the specifications for such software. With respect to products or software or partial assembly of products furnished by Dialogic but not manufactured by Dialogic, Dialogic hereby assigns to Customer, to the extent permitted, the warranties given to Dialogic by its vendors of such items.

B. If, under normal and proper use, a defect or non conformity appears in warranted products or software during the applicable Warranty Period and Customer promptly notifies Dialogic in writing during the applicable warranty period of such defect or non conformance, and follows Dialogic's instructions regarding return of such defective or non conforming Product or Software, then Dialogic will, at no charge to Customer, either:

- (i) repair, replace or correct the same at its manufacturing or repair facility or
- (ii) if Dialogic determines that it is unable or impractical to repair, replace or correct the product or software, provide a refund or credit not to exceed the original purchase price or license fee.

C. No product or software will be accepted for repair or replacement without the written authorization of and in accordance with instructions from Dialogic. Removal and reinstallation expenses as well as transportation expenses associated with returning such product or software to Dialogic shall be borne by Customer. Dialogic shall pay the costs of transportation of the repaired or replaced product or software to the destination designated in the original Order. If Dialogic determines that any returned product or software is not defective, Customer shall pay Dialogic's costs of handling, inspecting, testing and transportation. In repairing or replacing any product, part of product, or software medium under this warranty, Dialogic may use new, remanufactured, reconditioned, refurbished or functionally equivalent products, parts or software media. Replaced products or parts shall become Dialogic's property.

D. Dialogic makes no warranty with respect to defective conditions or non conformities resulting from any of the following: Customer's modifications, misuse, neglect, accident or abuse; improper wiring, repairing, splicing, alteration, installation, storage or maintenance performed in a manner not in accordance with Dialogic's or its vendor's specifications, or operating instructions; failure of Customer to apply Dialogic's previously applicable modifications or corrections; or items not manufactured by Dialogic or purchased by Dialogic pursuant to its procurement specifications. Dialogic makes no warranty with respect to products which have had their serial numbers removed or altered; with respect to expendable items, including, without limitation, fuses, light bulbs, motor brushes and the like; or with respect to defects related to Customer's data base errors. Improper packaging of product for repair will not be covered under this warranty agreement. No warranty is made that software will run uninterrupted or error free.

E. Warranty does not include:

- a) Dialogic's assistance in diagnostic efforts;
- b) access to Dialogic's Technical Support web sites, databases or tools;
- c) product integration testing;
- d) on-site assistance; or
- e) product documentation updates.

These services are available either during or after warranty at Dialogic's published prices.

F. THE FOREGOING WARRANTIES ARE EXCLUSIVE & ARE GRANTED IN LIEU OF ALL OTHER EXPRESS & IMPLIED WARRANTIES (WHETHER WRITTEN, ORAL, STATUTORY OR OTHERWISE), INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. CUSTOMER'S SOLE AND EXCLUSIVE REMEDY AND DIALOGIC'S SOLE OBLIGATION HEREUNDER, SHALL BE TO REPAIR, REPLACE, CREDIT OR REFUND AS SET FORTH ABOVE.

G. IN NO EVENT SHALL DIALOGIC, ITS DIRECTORS, OFFICERS, EMPLOYEES, AGENTS OR AFFILIATES, BE LIABLE FOR ANY COSTS OR DAMAGES ARISING DIRECTLY OR INDIRECTLY FROM YOUR USE OF ANY PRODUCT INCLUDING ANY INDIRECT,

INCIDENTAL, SPECIAL, EXEMPLARY, MULTIPLE, PUNITIVE OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, WHETHER BASED ON CONTRACT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY OR OTHER LEGAL THEORY, EVEN IF DIALOGIC, OR ANY OF ITS DIRECTORS, OFFICERS, EMPLOYEES, AGENTS OR AFFILIATES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY EVENT, DIALOGIC'S CUMULATIVE LIABILITY TO YOU FOR ANY AND ALL CLAIMS RELATING TO THE USE OF ANY PRODUCT SHALL NOT EXCEED THE TOTAL AMOUNT OF THE PURCHASE PRICE OR LICENSE FEES PAID TO DIALOGIC FOR SUCH PRODUCT.

H. CUSTOMER AND DIALOGIC HEREBY WAIVE THEIR RIGHT TO TRIAL BY JURY TO THE FULLEST EXTENT PERMITTED BY LAW IN CONNECTION WITH ALL CLAIMS ARISING OUT OF OR RELATED TO THIS WARRANTY, THE PRODUCTS COVERED HEREBY OR THE PERFORMANCE OF ANY PARTY HEREUNDER.

I. THIS WARRANTY SHALL BE CONSTRUED UNDER AND GOVERNED BY THE LAWS OF THE COMMONWEALTH OF MASSACHUSETTS WITHOUT GIVING EFFECT TO ANY CHOICE OR CONFLICT OF LAW PROVISION OR RULE (WHETHER OF THE COMMONWEALTH OF MASSACHUSETTS OR ANY OTHER JURISDICTION) THAT WOULD CAUSE THE APPLICATION OF THE LAWS OF ANY JURISDICTION OTHER THAN THE COMMONWEALTH OF MASSACHUSETTS. CUSTOMER SPECIFICALLY AND IRREVOCABLY CONSENTS TO THE PERSONAL AND SUBJECT MATTER JURISDICTION AND VENUE OF THE FEDERAL AND STATE COURTS OF THE COMMONWEALTH OF MASSACHUSETTS AND SUCH COURTS SHALL HAVE EXCLUSIVE JURISDICTION WITH RESPECT TO ALL MATTERS CONCERNING THIS WARRANTY OR THE ENFORCEMENT OF ANY OF THE FOREGOING.

J. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM JURISDICTION TO JURISDICTION.

About this Publication

Purpose

This publication provides guidelines for using the Dialogic® CSP.

Safety Labels

The following Safety labels may appear in this information product to alert customers to avoidable hazards. The following are in the order of priority:



DANGER

Danger indicates the presence of a hazard that will cause death or severe personal injury if the hazard is not avoided.



WARNING

Warning indicates the presence of a hazard that can cause death or severe personal injury if the hazard is not avoided.



CAUTION

Caution indicates the presence of a hazard that will or can cause minor personal injury or property damage if the hazard is not avoided. Caution can also indicate the possibility of data loss, loss of service, or that an application will fail.

Conventions used

This information product uses the text conventions explained below. In addition, hexadecimal numbers are preceded by a zero and small “x.” For example, the decimal number 15 is represented in hexadecimal as 0x0F.

Convention	Description
...	A horizontal ellipsis in an API message indicates fields of variable length.
:	A vertical ellipsis in an API message indicates that a block of information is repeated or is variable.
<i>n</i>	The letter <i>n</i> is a generic placeholder for a number.
Sans serif mono space	Indicates a command name, option, input, output, non-GUI error, and system messages.
<i>Sans serif monospace italic</i>	Indicates a parameter name in an input message. Example: move *.dot a: c: -s The -s is the parameter.
<i>Serif italic</i>	Indicates the name of a book, chapter, path, file, or API message. Example: <i>UserDirectory/Config.exe</i>
Boldface	Indicates keyboard keys, key combinations, and command buttons Example: Ctrl+Alt+Del
Sans serif boldface	Identifies text that is part of a graphical user interface (GUI). Example: Go to the Configuration menu and select Card->Span Configuration

Numerical List of EXS API Messages

0x0000	Connect	0x0000
0x0001	Channel Connection Status Query	0x0001
0x0002	Version Request	0x0002
0x0003	Connect With Pad	0x0003
0x0005	Connect With Data	0x0005
0x0006	Synchronization Priority List Query	0x0081
0x0007	Card Population Query	0x0007
0x0008	Release Channel	0x0008
0x000A	Service State Configure	0x000A
0x000B	Reset Configuration	0x000B
0x000C	Clear System Software	0x000C
0x000D	Subrate Connection Management	0x000D
0x000F	DS3 Configure/Query	0x000F
0x0010	Assign Logical Node ID	0x0010
0x0011	Trunk Type Configure	0x0011
0x0012	Filter/Timer Configure	0x0012
0x0013	Start Dial Configure	0x0013
0x0014	Transmit Signaling Configure	0x0014
0x0015	Receive Signaling Configure	0x0015
0x0016	Flash Timing Configure	0x0016
0x0017	Connect Wait	0x0017
0x0018	Busy Out	0x0018
0x0019	J1 Span Configure	0x0019
0x001A	Cross Connect Channel	0x001A
0x001B	Cross Disconnect Channel	0x001B
0x001C	Cross Connect Span	0x001C
0x001D	Cross Disconnect Span	0x001D
0x001E	Disconnect Tone Pattern	0x001E
0x0020	Outpulse Digits	0x0020
0x0021	Local End Release Mode Configure	0x0021

0x0023 Standby Line Card Configure 0x0023
 0x0024 Line Card Switchover 0x0024
 0x0028 Impulsing Parameters Configure 0x0028
 0x0029 Inseize Instruction List Configure 0x0029
 0x002A..... Outseize Instruction List Configure 0x002A
 0x002B..... Inseize Control 0x002B
 0x002C..... Outseize Control 0x002C
 0x002D..... Request For Service With Data 0x002D
 0x002E..... Call Processing Event 0x002E
 0x002F Connect Tone Pattern 0x002F
 0x0030 Transmit Cadence Pattern Configure 0x0030
 0x0031 Tone Configure 0x0031
 0x0034 Call Progress Analysis Result 0x0034
 0x0036 Release Channel With Data 0x0036
 0x0037 Channel Release Request 0x0037
 0x003A..... TFTP Manage 0x003A
 0x003D..... Generic Event Logger Configure/Query 0x003D
 0x0040 Request For Service 0x0040
 0x0042 DSO Status Change 0x0042
 0x0043 PPL Event Indication 0x0043
 0x0044 PPL Event Request 0x0044
 0x0045 Diagnostics Indication 0x0045
 0x0046 Generic Report 0x0046
 0x0047 CPC Detection 0x0047
 0x0049 Channel Released 0x0049
 0x004A..... Loop Timing Configure 0x004A
 0x004B..... Conference Create 0x004B
 0x004C..... Conference Delete Request 0x004C
 0x004D..... Conference Deleted 0x004D
 0x004E..... Connect To Conference 0x004E
 0x004F Connect One-Way To Conference 0x004F
 0x0050 Connect One-Way Forced 0x0050
 0x0052 Recorded Announcement Download Initiate 0x0052

0x0053 Recorded Announcement Download 0x0053
 0x0054 Recorded Announcement Delete 0x0054
 0x0055 Recorded Announcement Connect 0x0055
 0x0056 Recorded Announcement Disconnect 0x0056
 0x0057 Recorded Announcement Query 0x0057
 0x0058 Recorded Announcement Report 0x0058
 0x0059 Transmit Cadence Pattern Query 0x0059
 0x005A Tone Query 0x005A
 0x005B CCS Redundancy Configure 0x005B
 0x005C SS7 Signaling Stack Configure 0x005C
 0x005D SS7 Signaling Link Set Configure 0x005D
 0x005E SS7 Signaling Link Configure 0x005E
 0x005F SS7 Signaling Route Configure 0x005F
 0x0060 ISDN Interface Configure 0x0060
 0x0062 ISDN Terminal Configure 0x0062
 0x0063 ISDN Query 0x0063
 0x0064 SS7 Signaling Link Set Query 0x0064
 0x0065 SS7 Signaling Link Query 0x0065
 0x0066 SS7 Signaling Route Query 0x0066
 0x0067 SS7 CIC Query 0x0067
 0x0068 SS7 ISUP Message Query 0x0068
 0x0069 Channel Released With Data 0x0069
 0x006A SS7 CIC Configure 0x006A
 0x006B SS7 ISUP Message Format Configure 0x006B
 0x006C CCS Redundancy Query 0x006C
 0x006D SS7 Signaling Stack Query 0x006D
 0x006E Assign EXS Host/Slave 0x006E
 0x006F Node Status Query 0x006F
 0x0070 Node Status Report 0x0070
 0x0071 Ring Status Query 0x0071
 0x0072 Ring Status Report 0x0072
 0x0073 CCS Redundancy Report 0x0073
 0x0074 EXNET Ring Configure 0x0074

0x0077SS7 SCCP/TCAP Configure 0x0077
 0x0078SS7 SCCP/TCAP Query 0x0078
 0x0079Product License Download 0x0079
 0x007A.....Product License Query 0x007A
 0x007B.....V5 Configure 0x007B
 0x007C.....V5 Configuration Query 0x007C
 0x007D.....Matrix Configure 0x007D
 0x007E.....EXS Node Configuration Query 0x007E
 0x007FEXS Node Configure 0x007F
 0x0080Channel Parameter Query 0x0080
 0x0081Synchronization Priority List Query 0x0081
 0x0082System Log Query 0x0082
 0x0083Card Status Query 0x0083
 0x0084Span Status Query 0x0084
 0x0085T1 Span Query 0x0085
 0x0086Fault Log Query 0x0086
 0x0087Call Control Instructions Query 0x0087
 0x0089Impulsing Parameters Query 0x0089
 0x008A.....Call Progress Analysis Configuration Query 0x008A
 0x008E.....System Resource Usage Query 0x008E
 0x008FSS7 TUP Message Format Configure 0x008F
 0x0090SS7 TUP Message Query 0x0090
 0x0092SNMP Configure 0x0092
 0x0091Loop Back Configure/Query 0x0091
 0x0097Matrix Query 0x0097
 0x009B.....Download Begin BRecord 0x009B
 0x009C.....Download BRecord 0x009C
 0x009D.....Reset Matrix 0x009D
 0x009E.....Poll Request 0x009E
 0x009FPoll Interval Configure 0x009F
 0x00A1.....Become Active 0x00A1
 0x00A2.....Download Begin SRecord 0x00A2
 0x00A3.....Download SRecord 0x00A3

0x00A4..... Download Complete 0x00A4
 0x00A6..... Card Status Report 0x00A6
 0x00A8..... Assign Logical Span ID 0x00A8
 0x00A9..... T1 Span Configure 0x00A9
 0x00AB.....Poll 0x00AB
 0x00AF..... System Configuration 0x00AF
 0x00B2..... Call Progress Analysis Pattern Configure 0x00B2
 0x00B3.....Call Progress Analysis Class Configure 0x00B3
 0x00B4.....System Configuration Query 0x00B4
 0x00B5..... Time Set 0x00B5
 0x00B8..... Distant End Release Mode 0x00B8
 0x00B9.....Alarm 0x00B9
 0x00BA..... Generate Call Processing Event 0x00BA
 0x00BB..... Answer Supervision Mode Configure 0x00BB
 0x00BC.....Collect Digit String 0x00BC
 0x00BD..... DSP Service Request 0x00BD
 0x00BE..... DSP Service Cancel 0x00BE
 0x00BF..... Park Channel 0x00BF
 0x00C0.....DSP SIMM Configure 0x00C0
 0x00C1..... Alarm Cleared 0x00C1
 0x00C4..... D Channel Assign 0x00C4
 0x00C5.....D Channel De-assign 0x00C5
 0x00C6..... D Channel Facility List Configure 0x00C6
 0x00C8.....B Channel Configure 0x00C8
 0x00CA..... B Channel Query 0x00CA
 0x00CB.....D Channel Facility List Query 0x00CB
 0x00CD..... Span Filter Configure 0x00CD
 0x00CE.....Span Filter Query 0x00CE
 0x00CF.....PPL Timer Configure 0x00CF
 0x00D0..... Tag Configuration 0x00D0
 0x00D1.....PPL Assign 0x00D1
 0x00D2..... PPL Transmit Signal Configure 0x00D2
 0x00D3..... Busy Out Flag Configure 0x00D3

0x00D4..... PPL Create 0x00D4
 0x00D5.....PPL Table Download Initiate 0x00D5
 0x00D6..... PPL Table Download 0x00D6
 0x00D7.....PPL Configure 0x00D7
 0x00D8..... E1 Span Configure 0x00D8
 0x00D9..... PCM Encoding Format Configure 0x00D9
 0x00DA..... PPL Delete 0x00DA
 0x00DB.....E1 Span Query 0x00DB
 0x00DC.....PPL Audit Configure 0x00DC
 0x00DD.....PPL Audit Query 0x00DD
 0x00DE.....PPL Data Query 0x00DE
 0x00DF..... PPL Protocol Query 0x00DF
 0x00E0..... Virtual Card Configure 0x00E0
 0x00E2.....Virtual Span Control 0x00E2
 0x00E3.....Resource Attribute Configure 0x00E3
 0x00E4.....Resource Attribute Query 0x00E4
 0x00E5.....Matrix Status Report 0x00E5
 0x00E6..... IP Address Query 0x00E6
 0x00E7..... IP Address Configure 0x00E7
 0x00E8..... Route Control 0x00E8
 0x00E9..... Multi-Host Configure 0x00E9
 0x00EA.....SS7 CLI Configure 0x00EA
 0x00EB..... SS7 CLI Query 0x00EB
 0x00EE..... VoIP Protocol Configure 0x00EE
 0x00EF..... VoIP Protocol Query 0x00EF
 0x00F0.....IP Socket Configure 0x00F0
 0x00F1.....IP Socket Query 0x00F1
 0x00F2..... IP Socket Status Report 0x00F2
 0x00FB..... ARP Cache Report 0x00FB
 0x00FC..... ARP Cache Query 0x00FC
 0x0100.....IP Signaling Series 3 Card Configure 0x0100
 0x0101..... IP Signaling Series 3 Card Query 0x0101
 0x0102.....Recorded Announcement File System Query 0x0102

0x0103Recorded Announcement File System Defragment 0x0103
 0x0104 IP Signaling Series 3 Card Host Poll 0x0104
 0x0105 IP Signaling Series 3 Card Status Report 0x0105
 0x0107 Reset IP Signaling Series 3 Card 0x0107
 0x0108 Clear IP Signaling Series 3 Card 0x0108
 0x0117 Recorded Announcement Single Delete 0x0117
 0x0118Recorded Announcement File System Convert 0x0118
 0x0119 Recorded Announcement File System Report 0x0119
 0x011A..... DSP Cache Modify 0x011A
 0x011B..... Play File Start 0x011B
 0x011C.....Play File Modify 0x011C
 0x011D..... Play File Stop 0x011D
 0x011ERecord File Start 0x011E
 0x011FRecord File Modify 0x011F
 0x0120 Record File Stop 0x0120
 0x0121 Statistics Query 0x0121
 0x0122 Generic Card Configure 0x0122
 0x0123 Generic Card Query 0x0123
 0x0124 Resource Create 0x0124
 0x0125Resource Modify 0x0125
 0x0126 Resource Delete 0x0126
 0x0127 Resource Connect 0x0127
 0x0128Resource Disconnect 0x0128
 0x0129 Resource Delete Indication 0x0129
 0x012C..... Resource Query 0x012C
 0x012D..... Resource Release Request 0x012D
 0x012EResource Release Indication 0x012E
 0x0130NGA Configure 0x0130
 0x0131NGA Configure Query 0x0131
 0x0141 Resource Information Notify 0x0141
 0x0160NGA Service Configure 0x0160
 0x0161NGA Service Query 0x0161
 0x0162NGA State Query 0x0162

0x0163NGA State Notify 0x0163

EXS API Message Set

Alarm 0x00B9
Alarm Cleared 0x00C1
Answer Supervision Mode Configure 0x00BB
ARP Cache Query 0x00FC
ARP Cache Report 0x00FB
Assign EXS Host/Slave 0x006E
Assign Logical Node ID 0x0010
Assign Logical Span ID 0x00A8
B Channel Configure 0x00C8
B Channel Query 0x00CA
Become Active 0x00A1
Busy Out 0x0018
Busy Out Flag Configure 0x00D3
Call Control Instructions Query 0x0087
Call Processing Event 0x002E
Call Progress Analysis Class Configure 0x00B3
Call Progress Analysis Configuration Query 0x008A
Call Progress Analysis Pattern Configure 0x00B2
Call Progress Analysis Result 0x0034
Card Population Query 0x0007
Card Status Query 0x0083
Card Status Report 0x00A6
CCS Redundancy Configure 0x005B
CCS Redundancy Query 0x006C
CCS Redundancy Report 0x0073
Channel Connection Status Query 0x0001
Channel Parameter Query 0x0080
Channel Release Request 0x0037
Channel Released 0x0049
Channel Released With Data 0x0069

Clear IP Signaling Series 3 Card 0x0108
Clear System Software 0x000C
Collect Digit String 0x00BC
Conference Create 0x004B
Conference Delete Request 0x004C
Conference Deleted 0x004D
Connect 0x0000
Connect One-Way Forced 0x0050
Connect One-Way To Conference 0x004F
Connect To Conference 0x004E
Connect Tone Pattern 0x002F
Connect Wait 0x0017
Connect With Data 0x0005
Connect With Pad 0x0003
CPC Detection 0x0047
Cross Connect Channel 0x001A
Cross Connect Span 0x001C
Cross Disconnect Channel 0x001B
Cross Disconnect Span 0x001D
D Channel Assign 0x00C4
D Channel De-assign 0x00C5
D Channel Facility List Configure 0x00C6
D Channel Facility List Query 0x00CB
Diagnostics Indication 0x0045
Disconnect Tone Pattern 0x001E
Distant End Release Mode 0x00B8
Download Begin BRecord 0x009B
Download Begin SRecord 0x00A2
Download BRecord 0x009C
Download Complete 0x00A4
Download SRecord 0x00A3
DS0 Status Change 0x0042
DS3 Configure/Query 0x000F

DSP Cache Modify 0x011A
DSP Service Cancel 0x00BE
DSP Service Request 0x00BD
DSP SIMM Configure 0x00C0
E1 Span Configure 0x00D8
E1 Span Query 0x00DB
EXNET Ring Configure 0x0074
EXS Node Configuration Query 0x007E
EXS Node Configure 0x007F
Fault Log Query 0x0086
Filter/Timer Configure 0x0012
Flash Timing Configure 0x0016
Generate Call Processing Event 0x00BA
Generic Card Configure 0x0122
Generic Card Query 0x0123
Generic Event Logger Configure/Query 0x003D
Generic Report 0x0046
Impulsing Parameters Configure 0x0028
Impulsing Parameters Query 0x0089
Inseize Control 0x002B
Inseize Instruction List Configure 0x0029
IP Address Configure 0x00E7
IP Address Query 0x00E6
IP Signaling Series 3 Card Configure 0x0100
IP Signaling Series 3 Card Host Poll 0x0104
IP Signaling Series 3 Card Query 0x0101
IP Signaling Series 3 Card Status Report 0x0105
IP Socket Configure 0x00F0
IP Socket Query 0x00F1
IP Socket Status Report 0x00F2
ISDN Interface Configure 0x0060
ISDN Query 0x0063
ISDN Terminal Configure 0x0062

J1 Span Configure 0x0019
Line Card Switchover 0x0024
Local End Release Mode Configure 0x0021
Loop Back Configure/Query 0x0091
Loop Timing Configure 0x004A
Matrix Configure 0x007D
Matrix Query 0x0097
Matrix Status Report 0x00E5
Multi-Host Configure 0x00E9
Node Status Query 0x006F
NGA Configure 0x0130
NGA Configure Query 0x0131
NGA Service Configure 0x0160
NGA Service Query 0x0161
NGA State Notify 0x0163
NGA State Notify 0x0163
Node Status Report 0x0070
Outpulse Digits 0x0020
Outseize Control 0x002C
Outseize Instruction List Configure 0x002A
Park Channel 0x00BF
PCM Encoding Format Configure 0x00D9
Play File Modify 0x011C
Play File Start 0x011B
Play File Stop 0x011D
Poll 0x00AB
Poll Interval Configure 0x009F
Poll Request 0x009E
PPL Assign 0x00D1
PPL Audit Configure 0x00DC
PPL Audit Query 0x00DD
PPL Configure 0x00D7
PPL Create 0x00D4

PPL Data Query 0x00DE
PPL Delete 0x00DA
PPL Event Indication 0x0043
PPL Event Request 0x0044
PPL Protocol Query 0x00DF
PPL Table Download 0x00D6
PPL Table Download Initiate 0x00D5
PPL Timer Configure 0x00CF
PPL Transmit Signal Configure 0x00D2
Product License Download 0x0079
Product License Query 0x007A
Receive Signaling Configure 0x0015
Record File Modify 0x011F
Record File Start 0x011E
Record File Stop 0x0120
Recorded Announcement Connect 0x0055
Recorded Announcement Delete 0x0054
Recorded Announcement Disconnect 0x0056
Recorded Announcement Download 0x0053
Recorded Announcement Download Initiate 0x0052
Recorded Announcement File System Convert 0x0118
Recorded Announcement File System Defragment 0x0103
Recorded Announcement File System Query 0x0102
Recorded Announcement File System Report 0x0119
Recorded Announcement Query 0x0057
Recorded Announcement Report 0x0058
Recorded Announcement Single Delete 0x0117
Release Channel 0x0008
Release Channel With Data 0x0036
Request For Service 0x0040
Request For Service With Data 0x002D
Reset Configuration 0x000B
Reset IP Signaling Series 3 Card 0x0107

Reset Matrix 0x009D
Resource Attribute Configure 0x00E3
Resource Attribute Query 0x00E4
Resource Connect 0x0127
Resource Create 0x0124
Resource Delete 0x0126
Resource Delete Indication 0x0129
Resource Disconnect 0x0128
Resource Information Notify 0x0141
Resource Release Request 0x012D
Resource Query 0x012C
Resource Release Indication 0x012E
Resource Modify 0x0125
Ring Status Query 0x0071
Ring Status Report 0x0072
Route Control 0x00E8
Service State Configure 0x000A
SNMP Configure 0x0092
Span Filter Configure 0x00CD
Span Filter Query 0x00CE
Span Status Query 0x0084
SS7 CIC Configure 0x006A
SS7 CIC Query 0x0067
SS7 CLI Configure 0x00EA
SS7 CLI Query 0x00EB
SS7 ISUP Message Format Configure 0x006B
SS7 ISUP Message Query 0x0068
SS7 SCCP/TCAP Configure 0x0077
SS7 SCCP/TCAP Query 0x0078
SS7 Signaling Link Configure 0x005E
SS7 Signaling Link Query 0x0065
SS7 Signaling Link Set Configure 0x005D
SS7 Signaling Link Set Query 0x0064

SS7 Signaling Route Configure 0x005F
SS7 Signaling Route Query 0x0066
SS7 Signaling Stack Configure 0x005C
SS7 Signaling Stack Query 0x006D
SS7 TUP Message Format Configure 0x008F
SS7 TUP Message Query 0x0090
Standby Line Card Configure 0x0023
Start Dial Configure 0x0013
Statistics Query 0x0121
Subrate Connection Management 0x000D
Synchronization Priority List Configure 0x0006
Synchronization Priority List Query 0x0081
System Configuration 0x00AF
System Configuration Query 0x00B4
System Log Query 0x0082
System Resource Usage Query 0x008E
T1 Span Configure 0x00A9
T1 Span Query 0x0085
Tag Configuration 0x00D0
TFTP Manage 0x003A
Time Set 0x00B5
Tone Configure 0x0031
Tone Query 0x005A
Transmit Cadence Pattern Configure 0x0030
Transmit Cadence Pattern Query 0x0059
Transmit Signaling Configure 0x0014
Trunk Type Configure 0x0011
V5 Configure 0x007B
V5 Configuration Query 0x007C
Version Request 0x0002
Virtual Card Configure 0x00E0
Virtual Span Control 0x00E2
VoIP Protocol Configure 0x00EE

VoIP Protocol Query 0x00EF

SwitchKit Message Set

ActivateMatrix
AddLLCCard
AddLLCNode
AddSS7TCAPCard
AdjustMessageTimeout
Alarm
AlarmCleared
AllInService
AllocateChannel
AllocateChannelGroup
AllOutOfService
AnswerSuperviseConfig
AppConnectionQuery
AppDescriptionData
AppPopulationQuery
ARPCacheQuery
ARPCacheReport
AssignEXSHostSlave
AssignNode
AssignSpan
AssociateChanGroup
BChannelConfig
BChannelQuery
BroadcastLoad
BusyOut
BusyOutFlagConfig
CallControlInstructionQuery
CallProcessingEvent
CancelUserTimer
CardPopulationQuery

CardStatusQuery
CardStatusReport
CCSRedundancyConfig
CCSRedundancyQuery
CCSRedundancyReport
ChannelConnectionStatusQuery
ChannelGroupContentsQuery
ChannelGroupPopulationQuery
ChannelParameterQuery
ChannelProblem
ChannelReleased
ChannelReleasedWithData
ChannelReleaseRequest
ClearAllChanGroups
ClearChanGroup
ClearLog
ClearSoftware
ClearSystemSoftware
CollectDigitString
ConferenceCreate
ConferenceDeleted
ConferenceDeleteRequest
ConfigChanGroup
ConfigStatusMsg
ConfigSwitch
Connect
ConnectionStatusMsg
ConnectOneWayForced
ConnectOneWayToConference
ConnectToConference
ConnectTonePattern
ConnectWait
ConnectWithData

ConnectWithPad
CPAClassConfig
CPAConfigQuery
CPAPatternConfig
CPAResult
CPCDetection
CreateConnection
CrossConnectChannel
CrossConnectSpan
CrossDisconnectChannel
CrossDisconnectSpan
CSAPPLConfig
CSAPPLTimerConfig
D Channel Assign
DChannelDeAssign
DChannelFacilityListConfig
DChannelFacilityListQuery
DeviceServerConfig
DeviceServerEx
DeviceServerQuery
DeviceServerStatusReport
DiagnosticsIndication
DisconnectTonePattern
DistantEndReleaseMode
DownloadBeginBRecord
Download Begin SRecord
DownloadBRecord
DownloadComplete
DownloadSRecord
DS0StatusChange
DS3GenericMessage
DSPCacheModify
DSPServiceCancel

DSPServiceRequest
DSPSIMMConfig
DynamicConfig
E1SpanConfig
E1SpanQuery
FaultLogQuery
FilterTimerConfig
FlashTimingConfig
ForceGroupState
GELConfigQuery
GenerateCallProcessingEvent
GenerateLogMsg
GenericCardConfigure
Generic Card Query
GenericLLCReport
GenericReport
HexTool
HostAlarm
IgnoreChanGroup
ImpulsingParametersConfig
ImpulsingParametersQuery
InseizeControl
InseizeInstrListConfig
InterAppMsg
IPAddressConfig
IPAddressQuery
IPCallServerConfig
IPCallServerQuery
IPSocketConfig
IPSocketQuery
IPSocketStatusReport
ISDNInterfaceConfig
ISDNQuery

ISDNTerminalConfig
J1SpanConfig
LineCardSwitchover
LLCControl
LLCQuery
LocalReleaseConfig
LoopBackConfig
LoopTimingConfig
MessageWrapper
MonitorChannel
MultiHostConfig
NodeConfig
NodeConfigQuery
NodeStatusQuery
NodeStatusReport
OutpulseDigits
Outseize Control
OutseizeInstrListConfig
ParkChannel
PCMEncodingConfig
PlayFileModify
PlayFileStart
PlayFileStop
Poll
PollIntervalConfig
PollRequest
PPLAssign
PPLAuditConfig
PPLAuditQuery
PPLConfig
PPLCreate
PPLDataQuery
PPLDelete

PPLEventIndication
PPLEventRequest
PPLInitiateDownload
PPLProtocolQuery
PPLTableDownload
PPLTimerConfig
PPLTool
PPLTransmitSignalConfig
ProductLicenseDownload
ProductLicenseQuery
RecAnnConnect
RecAnnDelete
RecAnnDisconnect
RecAnnDownload
RecAnnDownloadInitiate
RecAnnFSConvert
RecAnnFSDefrag
RecAnnFSQuery
RecAnnFSReport
RecAnnQuery
RecAnnReport
RecAnnSingleDelete
RecordFileModify
RecordFileStart
RecordFileStop
ReceiveSignalingConfig
RedundantAppPoolMembersQuery
RedundantAppPoolsQuery
RedundantAppQuery
RedundantAppStatusMsg
RedundantLLCRegister
RegisterAsRedundantApp
RegisterVirtualName

ReleaseCacheChannels
ReleaseChannel
ReleaseChannelWithData
Remove Channels From Group
RemoveLLCRoutingID
RequestChannelAck
RequestChannelMsg
RequestForService
RequestForServiceWithData
RequestStandbyPoll
ReselectPrimaryApp
ResetConfig
ResetMatrix
ResetStandbyMatrix
ResourceAttributeConfig
ResourceAttributeQuery
Resource Connect
Resource Create
Resource Delete
Resource Modify
RingConfig
RingStatusQuery
RingStatusReport
RouteControl
ServerStatusChange
ServiceStateConfig
SNMPConfig
SpanFilterConfig
SpanFilterQuery
SpanStatusQuery
SS7CICConfig
SS7CICQuery
SS7CLLConfig

SS7CLLIQuery
SS7ISUPMessageFormatConfig
SS7ISUPMessageQuery
SS7SCCPTCAPConfig
SS7SCCPTCAPQuery
SS7SignalingLinkConfig
SS7SignalingLinkQuery
SS7SignalingLinkSetConfig
SS7SignalingLinkSetQuery
SS7SignalingRouteConfig
SS7SignalingRouteQuery
SS7SignalingStackConfig
SS7SignalingStackQuery
SS7TUPMessageFormatConfig
SS7TUPMessageQuery
StandbyLineCardConfig
StartDialConfig
Statistics Query
SubrateConnectionManagement
SwitchBackFromStandby
SwitchMgrQuery
SynchPriorityConfig
SynchPriorityQuery
SystemConfig
SystemConfigQuery
SystemLogQuery
SystemResourceUtilQuery
T1SpanConfig
T1SpanQuery
TagConfig
TCAPMessageRegister
TFTPManage
TimeSet

TransferChanMsg
TransmitCadencePatternConfig
TransmitCadencePatternQuery
TransmitSignalingConfig
TransmitToneConfig
TransmitToneQuery
TrunkTypeConfig
UpgradeSoftwarePatch
UserTimer
V5Config
V5Query
VersionRequestQuery
VirtualSlotConfig
VirtualSpanControl
VoIPProtocolConfig
VoIPProtocolQuery
WatchChanGroup
WriteCfgFile

1 EXS & SwitchKit API Messages

Overview

Purpose This chapter provides an introduction to API messages and then lists the EXS Application Programming Interface (APIs) messages created by Dialogic. The messages are listed alphabetically. Within the messages are hyperlinks to messages modules (AIBs, ICBs, and TLVs) that are carried by the messages.

Introduction to API Messages

Overview **Messages and Responses**

Messages and Responses each contain a header and a checksum, and can also contain AIBs (Address Information Blocks), ICBs (Information Control Blocks), TLVs (Tag-Length-Value blocks), and data.

Data

Data can reside in its own Data field or within AIBs, ICBs, or TLVs.

Field

A Field corresponds to a single row in the main Message/Response table, and it can be either one byte (for example 0x1B) or two bytes long (for example 0x1B4D). In accordance with Big Endian formatting, when a field is two bytes long, the first byte (in this case 0x1B) is the Most Significant Byte (MSB) and the second byte (in this case 0x4D) is the Least Significant Byte (LSB).

The next figure, *EXS API Message / Response Table*, provides an overview of API messages and refers to the chapters containing the specific information.

Figure 1-1 EXS API Message / Response Table

Messages and Responses each contain a header, checksum, and AIB (Address Information Block). Messages and Responses can also contain ICBs (Information Control Blocks), TLVs, (Tag-Length-Value blocks) and data. Data can reside in its own Data field or within ICBs or TLVs. A Field corresponds to a single row in the main Message/Response table, and it can be either one byte (for example 0x1B) or two bytes long (for example 0x1B4D). In accordance with Big Endian formatting, when a field is two bytes long, the first byte (in this case 0x1B) is the Most Significant Byte (MSB) and the second byte (in this case 0x4D) is the Least Significant Byte (LSB).

A **Message** is sent from the host computer to the EXS CSP, or vice versa.

A **Response** is sent from the EXS CSP in reply to a Message from the host computer, or vice versa. If the Response comes from the EXS CSP, the AIBs, ICBs, TLVs, and Data field values might match those that were sent in the corresponding Message.

Message (white)	Response (gray)
Header	Header
<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <p style="text-align: center;">AIB</p> <p>An AIB contains one or more AEs. AIBs are listed in the API message and are fully defined in the AIBs chapter.</p> <div style="border: 1px solid black; width: 80px; margin: 5px auto; text-align: center;">AE</div> <div style="border: 1px solid black; width: 100px; margin: 5px auto; text-align: center;">AE (possible)</div> </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <p style="text-align: center;">ICB</p> <p>There are three types of ICB: Action (0x01) Data (0x02) and Extended Data (0x03). An ICB can contain TLVs, and ISDN ICBs can contain IEs. ICBs are listed in the API messages and are fully defined in the ICBs chapter.</p> <div style="border: 1px solid black; width: 100px; margin: 5px auto; text-align: center;">TLV (possible)</div> </div> <div style="border: 1px solid black; padding: 5px;"> <p style="text-align: center;">TLV</p> <p>A TLV can stand on its own in a message or it can be embedded in an ICB. A TLV can also contain one or more levels of nested TLVs. In this case, the TLV Count applies to only the "outermost" (highest level) TLVs, and not to the nested TLVs.</p> <p>TLVs are listed in the API messages and are fully defined in the TLVs chapter.</p> <div style="border: 1px solid black; width: 100px; margin: 5px auto; text-align: center;">Nested TLV (possible)</div> </div>	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <p style="text-align: center;">AIB</p> <div style="border: 1px solid black; width: 100px; margin: 5px auto; text-align: center;">AE</div> <div style="border: 1px solid black; width: 100px; margin: 5px auto; text-align: center;">AE (possible)</div> </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <p style="text-align: center;">ICB</p> <div style="border: 1px solid black; width: 100px; margin: 5px auto; text-align: center;">TLV (possible)</div> </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <p style="text-align: center;">TLV</p> <div style="border: 1px solid black; width: 100px; margin: 5px auto; text-align: center;">Nested TLV (possible)</div> </div> <div style="border: 1px solid black; padding: 5px;"> <p style="text-align: center;">Status</p> <p>Unlike a Message, a Response can have a Status field, but only if the Response comes from the EXS CSP. A Response from the host computer has no Status field.</p> <p>A value that appears in the Status field is named a Response Status Value (RSV). If an RSV is used by more than one message, it is defined in the Common Response Status Values chapter. An RSV that is unique to a specific Response is defined on the Response side of the Message/Response table itself.</p> </div>
Checksum	Checksum

Generic API Message Format

API messages and responses are frames, containing strings of paired characters. The characters are represented in hexadecimal notation. Each frame consists of a standard header, followed by a variable number of data bytes. The data is unique to each message.

All messages, whether initiated by the CSP or by the host, must have a corresponding Response. The message fields are shown on the left (white) side of the table and the response fields are shown on the right (gray) side.

In this example, standard header information (bytes 0–) is indicated by the bold outline.

Field Descriptions

This section describes the fields in the header of each API message and response.

Frame

The first field in every message is always the *Frame* field. It is a one-byte value that indicates the beginning of the frame. The value **0xFE** is reserved for this purpose alone.

Because the value 0xFE is reserved for the *Frame* field, you must never use this value in any other field. If you must send the value 0xFE within the contents of a message, you must encode it using an escape sequence.

Length

The second field is always the *Length* field. It is a value that indicates the number of bytes to follow, excluding the *Checksum*.

Important! The system calculates the message length before using an escape sequence. You must do the same.

To calculate the value of the *Length* field, count the bytes between, but not including, the *Length* field and the *Checksum* field. In the message below, the length is 5. You can see the value 5 in the second byte of the length field, represented in hexadecimal notation as **0x05**.

MESSAGE	
Byte	Field Description
0	Frame Character (0xFE)
1	Length, MSB (0x00)
2	Length, LSB (0x05)
3	Message Type, MSB

MESSAGE	
Byte	Field Description
4	Message Type, LSB
5	Reserved (0x00)
6	Sequence Number
7	Node ID
8	Checksum

Message Type

The third field is always the *Message Type* field. It is a value that identifies the format and particular information contained in the message or response.

Sequence Number

Use the Sequence Number field(s), Bytes 5 and 6, to match corresponding responses and messages. Because the EXS API protocol is asynchronous, the system may contain more than one outstanding message of the same type. When a response to one of these messages returns, the response would be useless if there were no way to trace it to its corresponding message.

The Sequence Number resolves this issue by initiating either an 8-bit or 16-bit sequence number in an EXS API message. The sender of the message (either the host or the CSP) assigns the sequence number.

To provide backward compatibility, the 16-bit sequence number is enabled using the Generic Card Configure 0x0122 message and can be queried using the Generic Card Query 0x0123 message. The Matrix/Host Sequence Number Size TLV (0x0640) is used to configure either the 8-bit or 16-bit sequence number.

- 8-Bit Sequence Number

An 8-bit sequence number in the Byte 6 field is the default configuration. In this configuration, the Byte 5 field should always be set to 0x00 in CSP and Host initiated messages. Any other value in this field is reset to 0x00 in the response message to the Host. An 8-bit sequence number is in the range of 0x00-0xFF.

- 16-bit Sequence Number

A 16-bit sequence number extends the 8-bit sequence number Byte 6 (LSB) field into the Byte 5 (MSB) field to create a 16-bit sequence number. A 16-bit sequence number will be in the range of 0x00-0xFFFF.

When the CSP Matrix Series 3 Card is configured to use the 16-bit sequence number, Bytes 5 and 6 will contain the sequence number for both Host and CSP-initiated messages.

- A 16-bit sequence number in a Host-initiated message will follow the message through the CSP and be returned in the response.
- A 16-bit sequence number in a CSP-initiated message will be passed to the Host and be expected in the Host response.

Important! The 16-bit Sequence Number does not support LAPD and EXNET Connect® .

AIB

Address Information Blocks (AIBs) are used to address system objects such as slots, spans, and channels. An AIB can contain one or more Address Elements (AEs) to accommodate a variety of addressing requirements. An Address Element is a group of data bytes that represent specific address information.

Status

Most responses contain a *Status* field that follows the header. If the CSP processes a message successfully, it returns a positive acknowledgment (ACK) value of 0x10 in this field.

If an error occurs while the CSP is processing the message, the CSP returns a negative acknowledgment (NACK) value that indicates the reason for the error. Negative acknowledgements identify problems such as call processing anomalies and configuration errors. A negative acknowledgment value can be anything other than 0x10, which is reserved for positive acknowledgements.

Data

The contents of the data bytes vary for every message and response. Please refer to individual messages for data definitions.

Message Format Documentation Conventions

Checksum

The value of the *Checksum* field is the sum of the values of all header and data bytes, except for the *Frame* field. The system calculates the checksum before it uses an escape sequence, if an escape sequence is required.

This section describes the conventions used in this publication for the following:

- Byte Numbering
- Variable Number of Fields
- Word-sized Fields
- Bit Masks
- Double-Line in Table

Byte Numbering

In this Reference, the actual number of bytes in a message field is shown only if the value never changes. For example, the number of bytes in the header field is shown because the header bytes will always be 0-.

Variable Number of Fields

The number of bytes in a series often changes (for example, with variable data) so it cannot be shown in this Reference. In these cases, the first field is defined in the Reference, but subsequent fields are represented by vertical dots, as shown in below.

12	Data
:	:

Word-size Fields

API messages frequently use multiple byte fields for the following:

- Timers
- Conference IDs
- PPL Event IDs
- PPL Component IDs

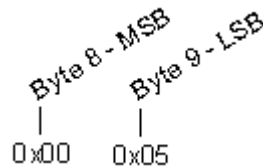
Fields that are more than one byte long use the Big Endian format of Most Significant Byte (MSB) first, Least Significant Byte (LSB) last.

For example, as shown in below, Byte 8 is the MSB and Byte 9 is the LSB for the *Conference ID* field.

Byte 8	Conference ID, MSB
Byte 9	Conference ID, LSB

If the Conference ID is 05, you would enter the ID as shown in the next figure. Note that the first value (MSB) is 0x00 until the value of the LSB field “overflows” (is greater than 255). These two-byte fields are often represented as a combined Hexadecimal number. For example, in the figure below, the combined representation of 0x00 and 0x05 would be **0x0005**.

Figure 1-2 Conference ID



Bit Mask

A bit mask is a byte that uses its individual bits to enable or disable functions. Setting the bit enables the function. This process is shown in the following example, taken from the *Alarm* message:

Example:

Data[1]Span Status

This is a bit mask indicating the information that is sent and received on the span. You can set multiple bits.

Bit

0 Receiving Red

1 Receiving Remote Alarm Indication/Yellow

2 Receiving Loss of Signal

3 Receiving Out of Frame

4 Sending Red (AIS)

5 Sending Remote Alarm Indication/Yellow

6 Receiving AIS

7 Receiving constant E-bit with Remote Alarm Indication. This bit is valid for E1 with insert CRC and Euro-ISDN.

When you print a message trace, you read a two-byte bit mask from right to left, as shown below (0 to 15).

15	. . .	8	7	. . .	0
----	-------	---	---	-------	---

Double Line in Table

A double line in a table indicates that you do not repeat the fields above it. In the example below, the double line between String 1 and String 2 indicates that you can have multiple strings in the ICB but that you do not repeat the information above the field, "String 2: BCD Digit Count."

ICB Type	0x02 (Data)
ICB ID	0x01
Data Length	Variable
Data[0]	Stage Number (1–4)
Data[1]	0x01 (Host-Supplied Digits)
Data[2]	Outputpulsing Signal Type 0x01 DTMF 0x02 MFR1 (host does not include KP or ST) 0x03 MFR2 0x04 MFR1 (host includes KP and any ST signal) 0x05 Dial Pulse
Data[3]	String Count (1 or 2)
Data[4]	String 1: BCD Digit Count
Data[5]	String 1: 1st BCD Digit Pair
:	:
:	String 1: Last BCD Digit Pair
:	String 2: BCD Digit Count
:	String 2: 1st BCD Digit Pair
:	:
:	String 2: Last BCD Digit Pair

Escape Sequences

The CSP uses the value 0xFE in the *Frame* field to indicate the beginning of a new frame. If a field value within a message or response is 0xFE, the CSP erroneously tries to process the next piece of data as a new message or response.

An escape sequence resolves this issue. Use the value 0xFD plus an extra byte that immediately follows it to send the values 0xFE or 0xFD. The table below shows the values that the system recognizes.

Value To Be Sent	Escape Sequence Values
0xFD	0xFD 0x02
0xFE	0xFD 0x01

When the CSP reads the value 0xFD within a frame, it uses the following byte to one-complement the 0xFD value. The sum of the complement byte and 0xFD produces the actual value sent.

Important! You cannot use the values 0xFE or 0xFD immediately following the initial 0xFD value.

When you calculate the length of a message that uses an escape sequence, do not include the extra byte that you added to the message. The length of a message is defined *before* the escape sequence is used.

For example, if you are sending the *Version Request* message and it includes the value 0xFE: The value of the *Length* field is the number of bytes between the *Length* and *Checksum* fields. The original length of this message is 0x. When you use the escape sequence, you add another byte to the message.

Although you added another byte to the message, you do not change the value of the *Length* field.

Frame	Length, MSB	Length, LSB	Message Type, MSB	Message Type, LSB	Reserved	Sequence Number	X	Logical Node ID	Checksum
FE	00	05	00	02	00	FD	01	01	05

Basic Message Set The primary function of the basic message set is to download the CSP Software from the host, and to query the CSP. When the CSP Software is completely downloaded, the CSP stops running from the CSP Matrix Series 3 Card firmware, and starts running from the CSP Matrix Series 3 Card RAM load. You can think of the Basic Message Set as a core group of messages.

The following messages comprise the basic message set:

- *Alarm*
- *Alarm Cleared*
- *Become Active*

- *Card Status Query*
- *Clear System Software*
- *Download Begin BRecord*
- *Download Begin SRecord*
- *Download BRecord*
- *Download Complete*
- *Download SRecord*
- *Fault Log Query*
- *Poll*
- *Poll Interval Configure*
- *Poll Request*
- *Reset Matrix*
- *TFTP Control/Query*
- *Version Request*

**API Messages Sent to the
Standby CSP Matrix Series
3 Card**

The host can send the following API messages to the standby matrix card.

- *Card Status Query (0x0083)*
- *Clear System Software (0x000C)*
- *Download Begin BRecord (0x009B)*
- *Download Begin SRecord (0x00A2)*
- *Download BRecord (0x009B)*
- *Download Complete (0x00A4)*
- *Download SRecord (0x00A3)*
- *Fault Log Query (0x0086)*
- *Poll (0x00AB) ACK*
- *Poll Interval Configure (0x009F)*
- *Poll Request (0x009E)*
- *Reset Matrix (0x009D)*
- *Version Request (0x0002)*

AddLLCCard

Type SwitchKit API message

Overview Use the *SK_AddLLCCard* message as an external application connected to the LLC. This message opens sockets to the SS7 cards on the node and sets up routing to the SS7 card.

AddLLCCard is used to create and connect pairs of card level device to the LLC. It has the ability to accommodate various card types including SS7 TCAP. Additional routing ID's can be added to the same connection by sending multiple *AddLLCCard* messages to the same IP pair and specifying a different Routing ID. By setting the action field to 0x01 (Remove Connection), it is assumed that all routing IDs on this connection will no longer be used and will therefore be removed by the LLC. For an application to maintain communication to the card, there must always be at least one routing ID configured for that IP pair.

This message cannot be used in a SwitchManager configuration file. Instead use the Add SS7TCAPCard message. SwitchManager will generate the appropriate *AddLLCCard* messages for you.

Sent by Application

C Structure

```
typedef struct {
    BaseFields Base;
    char CardIP1[30];
    char CardIP2[30];
    int RoutingID;
    int ControlNode;
    unsigned short CardType;
    UBYTE Action;
    UBYTE reserved88[5];
} SK_AddLLCCard;
```

C Structure Response

```
typedef struct {
    BaseFields Base;
    int Status;
    int RoutingID;
    UBYTE Reserved1;
    UBYTE reserved26[5];
} SK_AddLLCCardAck;
```

```

C++ Class class SKC_AddLLCCard : public SKC_ToolkitMessage {
public:
    SKC_AddLLCCard(int sz = 0);
    ~SKC_AddLLCCard();
    SK_DECLARE_CLASS(SKC_AddLLCCard,SKC_ToolkitMessage)

    virtual MsgStruct *getStructPtr();
    virtual const MsgStruct *getStructPtr() const;
    virtual int getTag() const;

    const char *getCardIP1() const;
    void setCardIP1(const char *x) ;
    const char *getCardIP2() const;
    void setCardIP2(const char *x);
    int getRoutingID() const;
    void setRoutingID(int x);
    int getControlNode() const;
    void setControlNode(int x);
    unsigned short getCardType() const;
    void setCardType(unsigned short x);
    UBYTE getAction() const;
    void setAction(UBYTE x);

```

```

C++ Class Response class SKC_AddLLCCardAck : public SKC_ToolkitAck {
public:
    SKC_AddLLCCardAck(int sz = 0);
    ~SKC_AddLLCCardAck();
    SK_DECLARE_CLASS(SKC_AddLLCCardAck,SKC_ToolkitAck)

    virtual MsgStruct *getStructPtr();
    virtual const MsgStruct *getStructPtr() const;
    virtual int getTag() const;

    int getStatus() const;
    void setStatus(int x);
    int getRoutingID() const;
    void setRoutingID(int x);
    UBYTE getReserved1() const;
    void setReserved1(UBYTE x);

```

AddLLCCardAck

The LLC will acknowledge AddLLCCard messages with an AddLLCCardAck message.

```
AddLLCCardAck(Status=<*Response Value>,
RoutingID=    < Unique ID for Card Pair Added>,
Reserved1=    <Not currently used>);
```

Status field Possible Response Values found in the status field of an AddLLCCardAck message.

*Response Values	Action
SK_SUCCESS	SubComponent connection was successfully added to the LLC.
SK_CONNECT_IN_USE	The RoutingID specified is currently in use.
SK_DUP_IP_ADDR	One of the IP's is being used by another pair of Devices.
SK_INTERNAL_LIMIT_REACHED	The max number of subComponents has been exceeded.
SK_INTERNAL_MEM_ERROR	There was a problem allocating memory within the LLC.
SK_NODE_DOESNT_EXIST	The node id specified has not been added to the LLC via an AddLLCNode message.
SK_BAD_IP_ADDRESS	An invalid IP string was encountered
SK_NO_MESSAGE	Invalid message parameter found while processing request.

RoutingID Macros To avoid duplicate values, the routing ID for each type of sub-component card must be derived using that cards corresponding get_RoutingID macro. Macros will be defined within SK_API.h so that they are globally accessible.

Macros available for deriving RoutingID's.

CardType	Function
SK_SS7TCAP_BOARD (0x03)	getSS7TCAP_RoutingID(int stackid(0-255));

AddLLCNode

Type EXS SwitchKit API message

For the same functionality, the EXS API message is Assign Node ID (0x0010).

Purpose Use *SK_AddLLCNode* at the initial configuration of your system and to dynamically add a node while Low-Level Communicator is running.

Important! To assign a logical node ID within a CSP system, the ring must be out-of-service (OOS). Assuming the ring is OOS and the nodes already assigned, if you try to re-assign a logical node ID, the *AssignNode* messages issued on behalf of the *AddLLCNode* message will be NACKed by that node. You must have unique logical node IDs at all times within a CSP system. For example, if you have two nodes (nodes 2 and 3) and you want to re-assign logical node ID 2 to node 3 and logical node ID 3 to node 2 the *AssignNode* messages will be NACKED by the CSP and the *AddLLCNode* will ultimately fail. You will have to de-assign logical node IDs 2 and 3 first by sending a *ResetConfig* message to each of the nodes. Only then can you freely assign node IDs using the *AddLLCNode* message.

Description The *AddLLCNode* message informs the LLC of a CSP Node. It specifies the Node ID and the CSP Matrix Series 3 Card IP addresses for each node. This can be used to add, update, or delete a node. The node must already have software downloaded through TFTP. Sending *SK_AddLLCNode* for a node that needs software will fail.



CAUTION

*Do not use the shared IP address that can be used for SIP and H.323 redundancy. See [Configuring a Shared IP Address Using BOOTP Server in the API Developer's Guide: Overview](#). Use the fixed IP addresses for the matrix cards in the *AddLLCNode()* message. If you don't, the LLC will inaccurately reflect the state of the CSP.*

Important! When LLC receives a request to connect to a node using *AddLLCNode()*, LLC will maintain that connection until a disconnect message is received. If the specified IP address is not

reachable, LLC will continue attempting the connection indefinitely. This condition will be logged in the LLC maintenance log file.

When the Switch Manager reconnects to the LLC, *AddLLCNode* messages are sent from the Switch Manager to the LLC. This causes the LLC to open a socket (connection) to the CSP and the *AssignNode* message (which is the switch-level equivalent of *AddLLCNode*) is sent to the CSP.

Sent by SwitchManager

Add a Node To add a node, you must send a configuration file with SwitchManager. The configuration file should contain at least the following information:

```
AddLLCNode (matrix1="10.10.55.10", matrix2="10.10.55.11",
             RequestedNode=0x08);
```

If you are adding a node with a single CSP Matrix Series 3 Card and you use the Matrix2 argument in your message, be sure to enter the 0-length string as its value. The configuration file should contain at least the following information:

```
AddLLCNode (matrix1="10.10.55.10", matrix2="",
             RequestedNode=0x08);
```

Arguments The following table shows the arguments you can change:

Argument	Description
Action	Specifies whether a node gets added, updated, or deleted. The values are: SK_ADD_NODE = 0 (default) SK_DELETE_NODE = 1 SK_UPDATE_NODE = 2
CallControlNode	Reserved
Matrix1	A text field corresponding to the IP address of the primary CSP Matrix Series 3 Card.
Matrix2	A text field corresponding to the IP address of the secondary CSP Matrix Series 3 Card, if it exists. If there is only one CSP Matrix Series 3 Card in the node, this field must be set to "", that is, the 0-length string.

Argument	Description
NodeType	<ul style="list-style-type: none"> • Needs to be specified for ISDN and H.323 device server nodes. NodeType = <i>SK_LSS_DS_SYSTEM</i> or 16. • Needs to be specified for EXNET Connect® nodes: NodeType = <i>SK_EXNET_SYSTEM</i> or 300. • If the above values are not used, the NodeType will default to SK_CS_SYSTEM or 200.
PhysicalNode	Reserved
RequestedNode	<p>RequestedNode is the logical node ID requested to be assigned to the new node. The API will attempt to assign this node:</p> <ul style="list-style-type: none"> • If there is no node ID currently on the node; • If it is currently assigned a different node ID. <p>It is not a guarantee that this node will be assigned. In particular, the node ID might already be in use on the ring. If so, AddLLCNode will fail, no connection will be established with the node. If the LLC is already connected to a node with the specified logical node ID, the AddLLCNode is considered an update, if treated accordingly.</p>
SeqNumWidth	<p>Determines whether a message will have two-byte sequence numbers or one-byte sequence numbers.</p> <p style="text-align: center;">A one-byte SeqNumWidth is 0xff A two-byte SeqNumWidth is 0xffff</p>
ISUPRemCon	<p>Specifies whether SS7 cards can act as though they are on a ring, even if they are not. The values are:</p> <p>If set to 0, a multi-node system must have an EXNET® (ring) card.</p> <p>If set to 1, forces the system to act as if it is a multi-node system even when there is no EXNET® (ring) card in the CSP.</p>

Configuration AddLLCNode (
Node = integer,
ConnectionID = integer,
Matrix1 = string,
Matrix2 = string,

AddLLCNode

```
RequestedNode = integer,  
Action = integer);
```

```
C Structure typedef struct {  
    char Matrix1[30];  
    char Matrix2[30];  
    int RequestedNode;  
    int PhysicalNode;  
    unsigned short NodeType;  
    unsigned short CallControlNode;  
    UBYTE Action;  
    unsigned short SeqNumWidth;  
    UBYTE ISUPRemCon;;  
    } SK_AddLLCNode;
```

C Structure Response

```
typedef struct {
    int Status;
    int ActualNode;
    UBYTE DownloadStatus;
} SK_AddLLCNodeAck;
```

C++ Class

```
class SKC_AddLLCNode : public SKC_ToolkitMessage {
public:
    const char *getMatrix1() const;
    void setMatrix1(const char *x);
    const char *getMatrix2() const;
    void setMatrix2(const char *x);
    int getRequestedNode() const;
    void setRequestedNode(int x);
    int getPhysicalNode() const;
    void setPhysicalNode(int x);
    unsigned short getNodeType() const;
    void setNodeType(unsigned short x);
    unsigned short getCallControlNode() const;
    void setCallControlNode(unsigned short x);
    UBYTE getAction() const;
    void setAction(UBYTE x);
    unsigned short getSeqNumWidth() const;
    void setSeqNumWidth(unsigned short x);
    UBYTE getISUPRemCon() const;
    void setISUPRemCon(UBYTE x);
};
```

C++ Class Response

```
class SKC_AddLLCNodeAck : public SKC_ToolkitAck {
public:
    int getStatus() const;
    void setStatus(int x);
    int getActualNode() const;
    void setActualNode(int x);
    UBYTE getDownloadStatus() const;
    void setDownloadStatus(UBYTE x);
};
```

AddSS7TCAPCard

Type EXS SwitchKit API message

Overview The *AddSS7TCAPCard* message is used to make a direct TCP connection between the LLC Host and a SS7 card for the purpose of routing TCAP traffic on a separate link. This message creates the appropriate AddLLCCard messages and sends them to the LLC.

Sent by Host, using Converged Services Administrator (CSA), webCSA or SwitchManager configuration file.

Arguments The following table shows the arguments that you can modify:

Argument	Description
CardIP1	The primary SS7 card IP address in the CSP.
CardIP2	The standby SS7 card IP address, if it exists. If there is only one SS7 card in the node, this field should be set to "", that is, the 0-length string.
StackID	The ID of the TCAP stack to which the LLC connects.
SSN	The Subsystem Number configured on the SS7 card to which the LLC should connect.
ControlNode	The logical node ID assigned to the CSP Matrix Series 3 Card which controls the chassis where the SS7 card resides.
Action	0x00 To add a connection 0x01 To remove a connection and all RoutingIDs

C Structure

```
typedef struct {
    char CardIP1[30];
    char CardIP2[30];
    UBYTE StackID;
    UBYTE SSN;
    int ControlNode;
    UBYTE Action;
    UBYTE reserved84[5];
} SK_AddSS7TCAPCard;
```

C++ Class `class SKC_AddSS7TCAPCard : public SKC_DummyMessage {`

```
public:  
    const char *getCardIP1() const;  
    void setCardIP1(const char *x);  
    const char *getCardIP2() const;  
    void setCardIP2(const char *x);  
    UBYTE getStackID() const;  
    void setStackID(UBYTE x);  
    UBYTE getSSN() const;  
    void setSSN(UBYTE x);  
    int getControlNode() const;  
    void setControlNode(int x);  
    UBYTE getAction() const;  
    void setAction(UBYTE x);
```

AdjustMessageTimeout

Type	EXS SwitchKit API message
Purpose	Use the <i>SK_AdjustMessageTimeout</i> message to change the default time-out for responses.
Description	<p>SwitchKit allows all requests to the CSP a fixed amount of time to complete before the request is considered timed out. For most messages, this time is 15 seconds which is often acceptable for these SwitchKit-defined time-outs. However, there are times when the time-out specified is insufficient. In particular, when several time consuming messages are sent to the same node simultaneously, the time to complete a request can increase due to latency within the CSP. These increased delays occur primarily during configuration time.</p> <p>The <i>AdjustMessageTimeout</i> message increases the time-out for all messages by <i>TimeoutIncr</i>. In other words, if an application requested that the time-out be adjusted by 10 seconds, the time-out on most messages sent by that application would become 25 seconds. Messages that have a larger default time-out would have their time-out increased by the same 10 seconds.</p> <p>You must set the <i>Action</i> field for <i>SK_AMT_ACTIVATE_TIMEOUT</i>, to activate a new time-out. To set this new time-out for all applications connected to the LLC, set the <i>NumApps</i> field to <i>SK_ALL_APPS</i>. If you attempt to set a new time-out for only one application, don't set the <i>NumApps</i> field.</p> <p>You can reset the time-out by setting the <i>Action</i> field to <i>SK_AMT_DEACTIVATE_TIMEOUT</i>.</p>
Sent by	Application
Arguments	The following table shows the user modifiable arguments of the <i>AdjustMessageTimeout</i> message:

Argument	Description
Action	Action specifies whether a new timeout gets activated or deactivated.
TimeoutIncr	TimeoutIncr specifies the amount of time added to the prior timeout value.
NumApps	NumApps specifies whether the new timeout gets set for only one or all messages.

Configuration *AdjustMessageTimeout* (
Node = integer,
ConnectionID = integer,
Action = integer,
TimeoutIncr = integer,
NumApps = integer);

C Structure typedef struct {
 UBYTE Action;
 unsigned short TimeoutIncr;
 UBYTE NumApps;
} *SK_AdjustMessageTimeout*;

C Structure Response typedef struct {
 int Status;
} *SK_AdjustMessageTimeoutAck*;

C++ Class class *SKC_AdjustMessageTimeout* : public
 SKC_ToolkitMessage {
public:
 UBYTE getAction() const;
 void setAction(UBYTE x);
 unsigned short getTimeoutIncr() const;
 void setTimeoutIncr(unsigned short x);
 UBYTE getNumApps() const;
 void setNumApps(UBYTE x);
};

C++ Class Response class *SKC_AdjustMessageTimeoutAck* : public SKC_ToolkitAck
 {
public:
 int getStatus() const;
 void setStatus(int x);
};

Alarm 0x00B9

SwitchKit Name Alarm

Type EXS API and SwitchKit API message

Description **Alarm 0x00B9**

The CSP sends Alarm 0x00B9 message to indicate that an alarm condition has been detected. The following information is reported:

- Severity
- Alarm Type
- Alarm Number
- Alarm Data

Alarm Cleared

In some cases, the CSP sends an *Alarm Cleared* message after the alarm condition ceases. The alarms that are followed by an *Alarm Cleared* message are marked with an asterisk (*).

Related Messages Alarm Cleared (AlarmCleared) 0x00C1

Sent by CSP

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE InfoSize;
    UBYTE Severity;
    UBYTE Entity;
    UBYTE AlarmNum;
    UBYTE Info[249];
} XL_Alarm;
```

C++ Class

```
class XLC_Alarm : public XLC_InboundMessage {
public:
    UBYTE getInfoSize() const;
    void setInfoSize(UBYTE x);
    UBYTE getSeverity() const;
    void setSeverity(UBYTE x);
    UBYTE getEntity() const;
    void setEntity(UBYTE x);
    UBYTE getAlarmNum() const;
    void setAlarmNum(UBYTE x);
    const UBYTE *getInfo() const;
    UBYTE *getInfo();
    void setInfo(UBYTE *x);
};
```

Resent in EXS API This message is resent once after five seconds.

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0005)
3, 4	Message Type (0x00B9)	3, 4	Message Type (0x00B9)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8	Checksum
8	Severity 0x01 Informative 0x02 Minor 0x03 Major		
9	Alarm Type 0x01 General Alarms 0x02 Card Alarms 0x03 Span Alarms 0x04 Channel Alarms 0x05 DSP SIMM Alarms 0x06 DSP Alarms 0x07 Node Alarms 0x08 DS3 Alarms 0x09 VoIP Module Alarms 0x0B SIP Alarm		
10	Alarm Number (see table below)		
11	Data Length (n)		
12	Data[0] (if applicable)		
:	Data[n-1]		
:	Checksum		

Alarm Type The table below contains a summary of the alarms showing the range of alarm numbers for each type and the length.

Alarm Information See the table for each alarm type below for the following information:

- Alarm Severity
- Alarm Number/Alarm
- Data Length
- Data

Summary of Alarms

Alarm Type	Alarm Number Value or Range	Data Length
0x01 General	0x01-0x1F, 0x21-0x22	Variable

Alarm Type	Alarm Number Value or Range	Data Length
0x02 Card	0x01-0x1C	Variable
0x03 Span	0x01-0x05	0x06
0x04 Channel	0x02	0x17
	0x03	0x12
0x05 DSP SIMM	0x01-0x06	0x06
0x06 DSP	0x01, 0x02	Variable
0x07 CSP Node	0x01-0x08, 0x0A-0x15	0x06
0x08 DS3	0x01 - 0x03	0x07
0x09 VoIP Module	0x00, 0x01, 0x02, 0x03	0x07

0x01 General Alarms

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Major	0x01*	System busy condition The CSP sends this alarm when the CSP Matrix Series 3 Card is flooded with messages. The CSP will ignore all incoming calls until the alarm is cleared.	0x00	The system ignores incoming calls until the alarm is cleared. Ignoring incoming calls decreases the traffic load and eventually the alarm will clear. Ensure that System Busy Warning messages are no longer being seen. Slowly ramp traffic back to normal levels and resume normal operation.

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
<p>Major: Default</p> <p>Minor: If clock mode was switched to REF1 or REF2.</p>	0x02	Clock mode switched	0x02	<p>The system automatically switches the clock mode to the next available clock source and reports the following data to the host:</p> <p>Data[0] - Current Synchronization Mode Data[1] - Previous Synchronization Mode</p> <p>Values for Data[0] and Data[1] fields:</p> <p>0x01 Primary Reference Clock Signal 0x02 Secondary Reference Clock Signal 0x03 Primary Loop Clock Signal 0x04 Secondary Loop Clock Signal 0x05 Free Running Clock Signal</p> <hr/> <p>This notification occurs when the clock mode was automatically switched or reference clock sources have been lost.</p> <p>The CSP will automatically set the clock mode to the next available clock source.</p> <p>Either a master timing source has been disconnected or gone bad, or a span signal has been lost.</p> <p>Investigate why the signal has been lost and try to restore it if possible. This signal may be controlled by the network in which the Loop timing is dependent.</p>
Inform.	0x03	Hardware cannot support DSP function		Verify the correct card type.
Major	0x04*	Reference 1 Clock Source Lost No host intervention is needed. The CSP will automatically switch the clock mode to the next available clock source.	0x00	<p>The system automatically switches the clock mode to the next available clock source.</p> <p>The Reference Clock Source has been lost. No host intervention is needed for this occurrence. The CSP will automatically switch the clock mode to the next available clock source.</p> <p>Have Operational Personnel Check to see what happened to the External Timing Source Connected the CSP Matrix I/O.</p>

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Major	0x05*	Reference 2 Clock Source Lost No host intervention is needed. The CSP will automatically switch the clock mode to the next available clock source.	0x00	<p>The system automatically switches the clock mode to the next available clock source.</p> <p>The Reference Clock Source has been lost. No host intervention is needed for this occurrence. The CSP will automatically switch the clock mode to the next available clock source.</p> <p>Have Operational Personnel Check to see what happened to the External Timing Source Connected the CSP Matrix I/O.</p>
Major	0x06	DSP Resource Blocked	0x01	Data[0] DSP Function Type
Major	0x06	DSP Resource Blocked	0x04	<p>Data[0] Slot where VRA Block occurred Data[1] Chain across SIMMs Data[2-3] RAN ID blocked</p> <hr/> <p>The CSP will send this alarm when all the resources of the particular resource type are all in use. The request to use a specific resource is placed in a queue until there is a resource of the alarmed type freed. The request is placed in a queue waiting for a resource to free unless a <i>DSP Service Cancel</i> (0x00BE) message is send by the host or the channel that the resource was requested on is release.</p> <p>See the DSP SIMM Configure (0x00C0) message in the API Reference and the <i>CSP Developer's Guide: Overview</i> for details on DSP function types.</p> <p>Either wait for the resource requested to free up which is the most likely response. Send a <i>DSP Service Cancel</i> (0x00BE) message. Release the channel that the resource was requested on. If you find that the CSP is reporting a large number of these alarm it maybe a good idea to re-evaluate the number of resources per function type you have configure. You may need to increase the number of configured functions type.</p>

Alarm 0x00B9

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Major	0x07	DSP Resource Function Type Not Configured	0x02	<p>Data[0] Resource Type Data[1] Resource Type 2 (0x00 if no second resource type)</p> <hr/> <p>The DSP function type requested is not configured and will need to be configured in order to use the resource being requested.</p> <p>Can also indicate a function is configured but the SIMM is out-of-service.</p> <p>See the DSP SIMM Configure (0x00C0) message in the <i>API Reference</i> and the <i>CSP Developer's Guide: Overview</i> for details on DSP function types.</p> <p>Configure the DSP function type for the resource being requested.</p>

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Major	0x08	<p>DSP Resource Management Inconsistent. (An internal task has asked to attach the same type of DSP resource to a single timeslot twice. This Alarm is sent when:</p> <ul style="list-style-type: none"> - A receiver has been requested but a Call Progress or Digit receiver for the same layer is already attached to the timeslot - A transmitter has been requested but a Digit or Recorded Announcement transmitter for the same layer is already attached to the timeslot 	0x09	<p>Data[0-6] 0x0D Channel AIB (Dialogic use only)</p> <p>Data[7] Process Virtual Identifier (The process that requested the DSP service. For example, Call Control, Layer 4, or E1 PPL) (Dialogic use only)</p> <ul style="list-style-type: none"> 06 - L4 PPL call processing 11 - Logical ID for all Layer 3's 1A - Layer 3 Plus 24 - SS7 25 - SS7 MTP2 28 - E1 Protocol Layer PPL 29 - T1 Protocol Layer PPL <p>Data[8] Resource Type (The DSP Resource requested. For example, DTMF Digit Transmitter).</p> <ul style="list-style-type: none"> 00 - Output CP 01 - Output digits 02 - Collect CP 03 - Collect digits 04 - Conference 05 - Output Announcement 06 - Detect energy 07 - Broadcast 08 - Collect FSK 09 - Output FSK 0A - Detect coin 0B - Play file 0C - Record file 0D - Modify play 0E - Fax XMT 0F - Fax RCV 10 - Modify record 11 - Echo cancel XMT 12 - Echo cancel RCV 13 - PVD/AMD RCV <hr/> <p>An internal task has asked to attach the same type of DSP resource to a single timeslot twice.</p> <p>Free the resource that is attached to the timeslot. Send down a <i>DSP Service Cancel</i> (0x00BE) message to release the channel and start a new call.</p>

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Inform.	0x09	Download Record Timeout	0x00	<p>Restart the download.</p> <p>This alarm only concerns legacy download procedures. The Series 3 Matrix card supports only Full TFTP. The host must follow each download message with another download message or a Download Complete message within five minutes, or the download aborts. For example, when a Download Begin BRecord is sent, a Download BRecord must follow within five minutes. The Download BRecord must be followed by either another Download BRecord or Download Complete within five minutes.</p> <p>Check ethernet connectivity or the application's download module for delays. Restart the download. If problem persists contact Dialogic Technical Support. Please be advised that an Ethernet capture may be requested.</p>
Major	0x09	Invalid Associated Channel	0x03	<p>Data [0] Logical Span ID MSB Data [1] Logical Span ID LSB Data [2] Channel</p> <hr/> <p>Verify the configuration and call flow in the socket.log file. If the problem persists, contact Dialogic Technical Support</p>
Inform.	0x0A	Partial Message Received (for example, incorrect checksum, zero length message, or message did not begin with frame character)	0x00	<p>Partial Message Received (for example, incorrect checksum, zero length message, or message did not begin with frame character)</p> <p>Collect the socket.log, alarm.log, maintenance_llc_xxx.log and a list of the applications running on the system and contact Dialogic Technical Support.</p>
	0x0B	RAN Requested is Unavailable in System		<p>Check if this announcement was reached in error. Check in the Vocabulary Index File to see if the file ID is listed there. If it is not listed, then report the problem to Dialogic Technical support along with socket.log, alarm.log, and timestamp of when the trap occurred.</p>

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Inform.	0x0C	Unacknowledged Message If you configured the system's message resend logic to "Resend Once and Generate <i>Alarm</i> ," these data fields contain the bytes of the resent message. For more information, refer to the <i>System Configuration</i> message.	Variable	Data[0-N] The bytes of the message Check for other error messages that would indicate a network or system condition leading to this problem.
Major	0x0D	Recorded Announcement ID Inconsistency	0x04	Data[0] Slot Number Data[1] SIMM Number Data[2, 3] RAN ID Sent upon completion of download or on card Insertion. This alarm indicates that an announcement being downloaded has the same ID as an existing announcement, but with different information. When the ID is specified in the <i>Recorded Announcement Connect</i> message, either announcement could be played. Each RAN must have a unique ID. One of the announcements (usually the older one) should be deleted. A Query addressed to one RAN ID returns information for only one of the RANs that share the ID. It is not possible to determine for which RAN the information will be returned. For example, if one file sharing the ID is 20 Kb and one file is 40Kb, only one of the sizes is reported, either 20 Kb or 40 Kb, but not both.
Minor	0x0E	SS7 Signaling Network Congestion When a route to a destination is congested, MTP sends the MTP_STATUS to ISUP which in turn sends this alarm to the host.	0x06	Data[0] SS7 Stack ID Data[1-4] DPC Data[5] Congestion Level Decrease the load or create a new route to the destination. If the system has not decreased the call load and this symptom persists, check the network engineering plan to possibly create a new route to the destination.

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Major	0x0F*	SS7 Remote ISUP Unavailable	0x05	<p>Data[0] SS7 Stack ID Data[1-4] DPC</p> <hr/> <p>SS7 Remote ISDN User Part (ISUP) unavailable. The address information represents the SS7 stack and Destination Point Code (DPC).</p> <p>Check the status of the Links and see if they are in service by sending the <i>SS7 Signalling Link Query</i> (0x0065) message. If they are not in service, attempt to determine why the links are Out Of Service. Make sure that they are marked in service from a host perspective using the <i>Service State Configure</i> (0x000A) message. Also check the status of the SS7 Signalling Routes using <i>SS7 Signalling Route Query</i> (0x0066). If the routes are unavailable, contact your SS7 service provider and ask why they are not in service.</p> <p>Also check the status of layer 1 span using <i>Span Status Query</i> (0x0084) message. This will tell you if the span is in sync, and if not how many slips and errors occurring on the physical span.</p>
Major	0x10	Node ID Already Exists	0x00	Check configuration and/or installation for correction.
Inform.	0x11	Invalid Message Length	0x00	<p>The application software sent a message with an invalid length field.</p> <p>Provide the application development team and Dialogic Technical support with the socket.log and the information from the trap.</p>

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Major	0x12	System Busy Warning	0x00	<p>This notification indicates that the traffic volume on the CSP is approaching its capacity. This message is only a warning. It does not mean a System Busy condition will definitely occur. The System Busy Warning does not have a clearing Alarm. It continues to occur every few seconds, until the CPU utilization reduces. The System Busy Warning messages can be between 2 and 20 seconds apart, depending on system load fluctuations.</p> <p>The EXS CSP CPU occupancy has exceeded the "System Busy Warning" threshold. This condition can happen due to traffic congestion or CSP configuration during high traffic or by having Debug printing enabled on a CSP card or cards.</p> <p>When a System Busy Warning is generated, the Host should reduce the number of new calls until the System Busy Warning messages cease. Whenever possible, ramp call rates up slowly and evenly to prevent an initial System Busy. Do not re-configure the CSP and do not enable unnecessary diagnostics.</p> <p>For more information on this alarm, refer to the CSP Developer's Overview Guide, Chapter 4 EXS API Application Development, section System Busy Warning and System Busy Alarm.</p>
Inform.	0x13	Resource Utilization Threshold Reached	0x01	<p>Data[0] Free Conferencing Timeslots (0x01)</p> <hr/> <p>This normally occurs during high traffic and high utilization of conference resources.</p> <p>Calculate the number of free timeslots available for EXS Conferencing using the System Resource Usage Query. Determine if free timeslots need to be made available. For complete information see CSP Conferencing Section in the <i>Configuring Multi-Node Systems of the Developer's Guide: Overview</i>.</p>

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
	0x14	Subrate Resources Unavailable		<p>The subrate resource (channels) provided by the SRC card are no longer available as (possibly) the card has been removed.</p> <p>Find out what the Subrate Card Status is and take actions depending on the result.</p> <p>Reset or change the card. If that does not work, contact Dialogic Technical Support.</p>
Major	0x15	SS7 Remote TUP Unavailable	0x05	<p>Data[0] SS7 Stack ID Data[1-4] DPC</p> <hr/> <p>Bring remote Telephone User Part in service.</p>
Major	0x16*	ISDN/SS7 Signaling Stack Congestion	0x03	<p>Data[0] Stack ID (SS7), Board Slot Number (ISDN)</p> <p>Data[1] Direction: 0x00 Incoming 0x01 Outgoing</p> <p>Data[2] Status: 0x00 Congestion Cleared (Alarm Cleared) 0x01 Congestion Level 1 0x02 Congestion Level 2 0x03 Congestion Level 3</p> <hr/> <p>Decrease the load on the ISDN/SS7 card. Consult Dialogic Technical Support.</p> <p>Calls will resume to normal after the congestion clears. You will receive an alarm cleared message when this occurs.</p>
Inform.	0x17	Route Table Updated	0x01	<p>Data[0] Table ID (1-10)</p> <hr/> <p>The active route table has been updated. The address information represents the table ID of the route table.</p> <p>Verify the route table is changed correctly.</p>

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Inform.	0x18	Resource Group Table Updated	0x01	Data[0] Table ID (1-10) <hr/> <p>The active resource group table has been updated. The address information represents the table ID of the route table.</p> <p>Verify the resource group table is changed correctly.</p>
Inform.	0x19	Old Route Table Deleted	0x01	Data[0] Table ID (1-10) <hr/> <p>A previously active routing table has been deleted. The address information represents the table ID.</p> <p>Verify that the table being deleted is no longer needed.</p>
Inform.	0x1A	Old Resource Group Table Deleted	0x01	Data[0] Table ID (1-10) <hr/> <p>A resource group table has been deleted.</p> <p>Verify the table being deleted is no longer needed.</p>
Inform.	0x1B	Route Table Forcefully Deleted	0x01	Data[0] Table ID (1-10) <hr/> <p>The active route table was forcefully deleted. The address information represents the table ID.</p> <p>Verify the table being deleted is no longer needed.</p>
Inform.	0x1C	Resource Group Table Forcefully Deleted	0x01	Data[0] Table ID (1-10) <hr/> <p>The active resource group table has been forcefully deleted. The address information represents the table ID.</p> <p>Verify the table being deleted is no longer needed.</p>

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Major or Minor (see Data)	0x1D	Flash Copy or Load Problem	0x02	<p>Data[0] Severity</p> <ul style="list-style-type: none"> 0x00 Major 0x01 Major 0x02 Major 0x03 Major 0x04 Minor 0x05 Minor 0x06 Major <p>Data(1) Entity</p> <p>Valid entries for this field based on the Entity field value 0x00 are as follows:</p> <ul style="list-style-type: none"> 0x00 Flash erase failed 0x01 Flash write failed 0x02 RAM load to copy is bad 0x03 Flash load bad after copy from RAM 0x04 RAM and flash have different timestamps 0x05 RAM load bad after a soft reset 0x06 RAM load bad after copy from Flash <hr/> <p>There is a problem during the flash or RAM loading.</p> <p>Contact Dialogic Technical Support.</p>

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Major	0x1E	TFTP Alarm	Variable	<p>Typically the following events result in this alarm.</p> <ol style="list-style-type: none"> 1. Filename/path that is returned in BOOTP Response does not match. 2. TFTP Server is not reachable or running. 3. TFTP Configuration File indicates a load that is not resident or incorrectly named on the TFTP Server. 4. The TFTP file has been altered from previous version. <p>There are four things to validate:</p> <ol style="list-style-type: none"> 1. Verify that TFTP is running on the machine with the software files. 2. Verify that the TFTP Server is a reachable ethernet address from the CSP. 3. Verify that the BOOTP response indicates the correct name and path of the tftp configuration file. 4. Verify that the tftp configuration for the all .bin files specified in the tftp configuration exist on the TFTP server and contain the correct path and filenames of the software .bin files. Assume Dialogic Technical support will require information on BOOTP, TFTP, and Ethernet setups. An ethernet trace may be required. Debug access to the Matrix may also be required. <p>See 0x1E TFTP Alarm after this table.</p>

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Major	0x1F	System Memory Low		<p>System memory has dropped below the minimum level of satisfactory performance. When the system is in this condition, it is also in a System Busy condition and the CSP sends a System Busy Alarm (0x01) to the host.</p> <p>Check for other error messages that would indicate a network or system condition leading to this problem.</p> <p>Check Ring Interface hardware and then bring the ring in service.</p>
Major	0x20	System Memory Low Cleared		<p>Indicates that the system returns to the minimum level of satisfactory performance. The CSP sends a System Memory Satisfactory Alarm (0x20) to the host, along with an Alarm Cleared (0xC1) message.</p> <p>Resume normal operation.</p>
Major	0x21	Host Connection Dropped	0x02	Data[0] Multi-host port ID 0x3142 - 0x3147
Major	0x22	EXNET Ring Down	0x08	Data[0-4] Slot AIB Data[5] Logical Ring ID Data[6] Host Service State Data[7] EXNET Ring Error Status <hr/> <p>This notification is sent when an EXNET® ring goes down. The address information represents the slot of the EXNET-ONE card and logical ring ID of the ring.</p>
Inform.	0x2F	Log F/Log B Buffer is 75% Full		
Inform.	0x30	Log T Buffer is 75% Full	0x01	Data[0] Slot Number <hr/> Contact Dialogic Technical Support.
Inform.	0x31	Log F/Log B Buffer is Full	0x01	Data[0] Slot Number <hr/> Contact Dialogic Technical Support.
Inform.	0x32	Log T Buffer is Full	0x01	Data[0] Slot Number <hr/> Contact Dialogic Technical Support.
	0x33	MGC Lost Connection		

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Inform.	0x34	DSP CPA Configuration Conflict	0x04	Data[0] Tone Group: 0x00 Data[1] Tone ID: 0x10 Data[2-3] Frequency 0x01-0xA4 Consult the <i>CSP Developers Guide: Overview</i> for details on configuring CPA. Modify the configuration.
Major	0x35	ISDN Congestion		
	0x3A	SS7 Signaling Destination Congestion Sent to the host when a particular remote node is congested as indicated by a change in the congestion level.	0x06	Data[0] Stack ID Data[1-4] DPC Data[5] ACL Level If the system has not decreased a call to this DPC and this symptom persists, check the network engineering plan to possibly create a new route to the destination.
Major	0x3B	Signaling Link Congestion When a link is congested, MTP sends MTP Status to ISUP which in turn sends this alarm to the host.	0x06	Data[0] Stack ID Data[1-4] DPC Data[5] LINK_CONGESTION_LEVEL If the system has not decreased a call to this DPC and this symptom persists, check the network engineering plan to possibly create another link in the link set.
Inform.	0x3D	NFS RAM Disk Configuration Failure	0x03	Data[0] Slot Data[1,2] Server ID The VIF is missing or wrong. Check NFS settings for the DSP2 Card.
Major	0x3F	Insufficient Resource Points for DSP Function	0x01	Data[0] Function Type The system application has requested more than the allowed resource points configured on the system. Increase the number of resource points in the system with a downloadable license. Contact your sales person to purchase more licenses. You can also contact your sales engineer who can help you design your DSP2 configuration as well as estimate the number of resource points needed.
Inform.	0x3E	NFS Server Disconnect	0x02	Data[0,1] Server ID
Major	0x40	DSP Request Queued Due to DSP Series 2 CPU Overload Level 2	0x01	Data[0] DSP Function Type Requested Either reduce recourse requests at the application layer or add more DSP2 cards to the system.

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Inform.	0x41	Channel has an Echo Canceller resource attached but the requested receiver resource is not available on the same DSP Card.	0x05	Data[0] – Slot of DSP Card providing Echo Cancel to channel Data[1] – Function type of requested receiver Data[2, 3] – Span (MSB, LSB) Data[4] - Channel When attaching echo cancellers together with DSP receivers, be sure that the same DSP2 card is being used. The host can force the slot AIB for both message.
Minor	0x42	DSP Series 2 FSK General Alarms	0xnn (Variable)	Data[0] FSK Alarm ID 0x01 FSK Stop ACK Not Rcvd from DSP Chip 0x02 Invalid Resource Type for SMS 0x03 Internal Error Setting PPL Timer Data[1,2] Midplane Time Slot Data[3] DSP Series 2 Slot Data[4,n] Additional Data Specific to FSK Alarm IDs
Major	0x43	DSP2 RTP	0x06	Data[0] RTP Alarm ID: 0x01 RTP socket open error while Record Data[1] DSP2 slot Data[2-5] File ID
Minor	0x44	SCCP/TCAP Capacity Exceeded	0x04	Data[0-3] Current "Leaky Bucket" value in TCAP transactions Minor alarm is generated with each one second of licensed bucket capacity exceeded. Major alarm generated once 10 seconds of bucket capacity is exceeded indicating all TCAP transactions are now being rejected. The alarm clears when the bucket capacity returns to a 0.5 second of licensed capacity.

*Followed by an *Alarm Cleared* message

0x02 Card Alarms

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
	0x01	Card Taken Out of Service	Variable	0x01 Slot AIB Check with operations personnel to verify card removal. Once card is brought in service, the CSP sends the <i>Alarm Cleared</i> message (0x00C1).
Major	0x02	Unable to Allocate Timeslots for this Card	0x05	0x01 Slot AIB Verify that the CSP contains the required number of available timeslots for this card. If sufficient timeslots are available, contact Dialogic Technical Support.
	0x03	Card Download is Corrupt	0x05	0x01 Slot AIB Reseat the card. Try a different card. Try a different software load. If all above options fail, contact Dialogic Tech Support.
Major	0x04	Card Type Not Recognized	0x05	0x01 Slot AIB Check the TFTP server and the tftp configuration file to make sure the card being used is included. Contact Dialogic Technical Support.
-	0x05	Reserved		
-	0x06	Reserved		
Inform.	0x07	Card's Configuration has been Reset to Defaults	0x05	0x01 Slot AIB One of the following occurred: Card has been reset by the host/switchkit. Card has reset/faulted due to software error. Card has reset due to a hardware problem. If running Switchkit, Switchmgr will take care of reconfiguring the card. If not using Switchkit the host has to reconfigure the card.
-	0x08	Reserved		
Major	0x09	Card Reset	0x05	0x01 Slot AIB Card has either faulted (software error) or was manually reset (either via software or via push button reset) The card should come back in service by itself.
	0x0A	Connection Map Failure	0x05	0x01 Slot AIB

Alarm 0x00B9

Major	0x0B	Software Does Not Have Download for Card	0x05	0x01 Slot AIB <hr/> Software cannot be downloaded to the card. Check the TFTP server and the tftp configuration file to make sure the card being used is included. Contact Dialogic Technical Support.
Major	0x0C	Card Does Not Support A-Law Encoding Format	0x05	0x01 Slot AIB <hr/> Card does not support A-law encoding format. Check the card type and the card configuration.
Major	0x0D	Card Revision Not Supported by System Software	0x05	0x01 Slot AIB <hr/> Card revision not supported by system software revision. Replace the card with a different REV card. The REV can be found on the card itself and is made of a letter and a two digit number (for example, C12). The higher letter/number the newer the REV.
Major	0x0E	Card Removed	0x05	0x01 Slot AIB <hr/> Check with operations personnel to verify card removal.
Major	0x0F	Hardware Failure Occurred on Slot Specified The CSP also sends a <i>Card Status Report</i> with a description of the failure	0x05	0x01 Slot AIB <hr/> A hardware failure has been detected on a card. Reseat the card. If that does not work, replace card.
Minor	0x10	Ethernet Host Link Error Detected NOTE: To use the Host Link Failure Detection feature and receive this message, you must have polling enabled.	0x00	
Major	0x11	Card or Fan Tray Not Installed in Chassis	0x05	0x01 Slot AIB <hr/> Check if fan tray and card are installed.

Major	0x12	EXNET Port A - Communication Failure	0x08	<p>Data[0-4] Slot AIB Data[5] Logical Ring ID Data[6] Host Service State Data[7] EXNET Ring Error Status</p> <hr/> <p>Hardware failure on EXNET Port A.</p> <p>The Remote Node terminating to the local EXNET® Port A is not accessible (that is, the problem is not with local node but with the remote node).</p> <p>Verify EXNET Port A's terminating connection (remote node) is accessible.</p> <p>There could be a faulty fiber cabled connected to EXNET Port A. Check fiber connections are snug and connected properly.</p> <p>Bring the ring out of service then back in service. Reset Ring Interface card.</p> <p>If the lights on the Ring Interface card are still not all green, then collect the socket.log,alarm.log, and maintenance_llc_xxx.log and contact Dialogic Technical Support.</p>
Minor	0x13	EXNET Port B - Communication Failure	0x08	<p>Data[0-4] Slot AIB Data[5] Logical Ring ID Data[6] Host Service State Data[7] EXNET Ring Error Status</p> <hr/> <p>Hardware failure on EXNET Port B</p> <p>The Remote Node terminating to the local EXNET® Port B is not accessible (that is, the problem is not with local node but with the remote node).</p> <p>There could be a faulty fiber cabled connected to EXNET® Port B. Check fiber connections are snug and connected properly.</p> <p>Verify EXNET® Port B's terminating connection (remote node) is accessible.</p> <p>Bring the ring out of service then back in service. Reset Ring Interface card.</p>
Minor	0x14	EXNET Port A - Fiber Laser Failure	0x05	<p>0x01 Slot AIB</p> <hr/> <p>Refer to the <i>Hardware Installation and Maintenance Guide</i>.</p>

Alarm 0x00B9

Minor	0x15	EXNET Port B - Fiber Laser Failure	0x05	<p>0x01 Slot AIB</p> <hr/> <p>The fiber driver on the Ring interface card has gone bad.</p> <p>Contact Dialogic Technical Support and replace the Ring Interface Controller card.</p>
	0x16	Subrate Card Switchover		A switchover has occurred. No action required.
Inform.	0x17	Software Error (Please report this error to Dialogic Technical Support)	Variable	<p>0x01 Slot AIB</p> <p>Data[0] Address Method (0x00)</p> <p>Data[1] Number of Address Elements (0x01)</p> <p>Data[2] Address Type (0x01)</p> <p>Data[3] Data Length (0x01)</p> <p>Data[4] Slot Number</p> <p>Data[5-44] Filename (in ASCII)</p> <p>Data[45,46] Line Number</p> <p>Data[47-n] Diagnostic Data</p> <hr/> <p>A software error occurred on a line card. The address information represents the slot containing the card.</p> <p>Gather information as to what was occurring on the system at the time of the error and report to Dialogic Technical Support.</p>
Major	0x18	Remote SS7 card removed	0x05	<p>0x01 Slot AIB (if SS7 card in local node)</p> <p>0x69 Slot Number (if SS7 card in remote node)</p> <hr/> <p>Check with operations personnel to verify card removal.</p> <p>Re-insert the working SS7 card and, Switch Manager will reconfigure this card.</p> <p>SwitchManager will trigger a <i>Card Status Report</i> (0x00A6) message and reconfigure the card per the configuration file that was used to start/configure the system.</p>
Major	0x19	Bus Contention with Rear Card	0x05	<p>0x01 Slot AIB</p> <hr/> <p>The front card or I/O card are mismatched and incompatible with each other.</p> <p>Verify the I/O and card type and make sure they are supposed to be paired up.</p>

Alarm 0x00B9

Major	0x1A	Bus Contention with Front Card	0x05	<p>0x01 Slot AIB</p> <hr/> <p>The front card or I/O card are mismatched and incompatible with each other.</p> <p>Verify the I/O and card type and make sure they are supposed to be paired up.</p>
Major	0x1B	Internal Diagnostic Failure	0x08	<p>Data[0-4] Slot AIB Data[5] Logical Ring ID Data[6] Host Service State Data[7] EXNET Ring Error Status</p> <hr/> <p>An internal diagnostic error has been detected on an EXNET Ring card.</p> <p>Test the Ring Interface Card in a different slot. If the matrix continues to report this alarm to the host, replace the Ring Interface Card.</p>
Major	0x1C	EXNET card Looped Out of Ring	0x08	<p>Data[0-4] Slot AIB Data[5] Logical Ring ID Data[6] Host Service State Data[7] EXNET Ring Error Status</p> <hr/> <p>Test the Ring Interface Card in a different slot. If the matrix continues to report this alarm to the host, replace the Ring Interface Card.</p>
Major	0x1D	Card Added to Virtual Slot	0x05	<p>0x01 Slot AIB</p> <hr/> <p>The host or Switchkit has configured a virtual card.</p> <p>Verify card type and that the card has been placed in the correct slot. If symptoms continue, contact Dialogic Technical Support for configuration guidance.</p>

Major	0x1E	Line Card Approaching Busy/ Line Card Approaching Busy Clear	Variable	<p>SS7 card Data[0] Slot Number Data[1-n] Reason Codes: 0x00 Low Memory 0x01 Low MCB 0x02 Low Idle CPU Data[n+1] Stack ID</p> <p>16-Span SLC Data[0] Slot Number Data[1-n] Reason Codes: 0x00 Low Memory 0x01 Low MCB 0x02 Low Idle CPU</p> <hr/> <p>Line card busy conditions exceeding the line card "Approaching Busy Threshold".</p> <p>Reduce the line card load by diverting traffic from the overloaded line card. Disable Debug if enabled on the line card.</p>
Major	0x1F	Line Card Busy Line Card Busy Clear	Variable	<p>SS7 card Data[0] Stack ID Data[1] Slot Number Data[2-n] Reason Codes: 0x00 Low Memory 0x01 Low MCB 0x02 Low Idle CPU</p> <p>16-Span SLC Data[0] Slot Number Data[1-n] Reason Codes: 0x00 Low Memory 0x01 Low MCB 0x02 Low Idle CPU</p> <hr/> <p>Caused by line card overload busy conditions.</p> <p>Reduce line card load by diverting traffic from the overloaded line card. Disable Debug if enabled on the line card.</p>
	0x20	DS3 Card Limited Spans for SS7	0x05	<p>0x01 Slot AIB</p> <hr/> <p>Look for other error messages that indicate a network or system condition leading to this problem. If symptoms persist contact Dialogic Technical Support.</p>

Alarm 0x00B9

	0x21	Too many DS3s Configured for Limited Spans	0x05	<p>0x01 Slot AIB</p> <hr/> <p>Ensure proper DS3 card DIP Switch positioning.</p> <p>Only one DS3 card can have DIP Switch 4 and 6 set to OFF "limited SS7 CIC traffic". The Standby DS3 card, must always have DIP switch 4 and 6 set to the ON position.</p>
Major	0x22	Ethernet Link Failure	0x06	<p>Data[0-4] Slot AIB (0x01) Data[5] I/O Card Ethernet Port Number</p> <p>This alarm is sent when an SS7 card loses its connection to a local host.</p> <hr/> <p>This alarm is sent if a VDAC-ONE I/O or Multi-Function Media I/O card port that was previously in a steady LINK_UP state loses its connection. The VDAC-ONE slot and failed I/O port (0-2 for the 3 I/O ports on the Multi-Function Media I/O card) is sent in the data portion of the Alarm message. The alarm is cleared when the link returns to a steady Link-Up state.</p> <p>This alarm is sent if any I/O ports lose their network connection, and will be cleared when the connection is re-established.</p> <p>Trace the network cable between the I/O card and the remote Ethernet switch. Replace the Ethernet cable or the connector if necessary.</p>
Inform.	0x39	Ethernet Link-Up	0x06	<p>Data[0-4] Slot AIB (0x01) Data[5] Ethernet Port Number (0x00-0x02)</p> <hr/> <p>This alarm is sent to the host the first time a link-up state is seen on the Multi-Function Media I/O card.</p> <p>Because there are three Ethernet ports on the Multi-Function Media I/O card, the host might see three of these alarms when the card comes into service.</p> <p>No action required.</p>
Inform.	0x3E	NFS Server Disconnect	0x07	<p>Data[0-4] Slot AIB (0x01) Data[5, 6] Server ID</p> <hr/> <p>The connection between the DSP2 card and the NFS server has broken down.</p> <p>Check the Ethernet network and all Ethernet connections. If the alarms persists, update the <i>Generic Card Configuration</i> message in the CSP configuration with a NFS Poll Retries (0x601) TLV. This TLV can change the default number of polls that can be missed before this alarm is sent.</p>
Inform.	0x40	NFS Server Mounted Successfully	0x07	<p>Data[0-4] Slot AIB (0x01) Data[5, 6] Server ID</p>

Alarm 0x00B9

Major	0x41	NFS Server Mounting Error	0x09	<p>Data[0-4] Slot AIB (0x01) Data[5, 6] Server ID Data[7, 8] Error ID</p> <hr/> <p>0x0001 - NFS Polling and Ping Failed 0x0002 - NFS Polling and Mount Failed 0x0003 - NFS Server Unreachable</p> <p>Diagnose the LAN connection between the DSP2 card and the NFS server.</p>
Inform.	0x42	DSP Series 2 File Space Low	0x0C	<p>Data[0-4] Slot AIB (0x01) Data[5, 6] Number of Files Data[7, 8] Number of Free Blocks Data[9] TFTP (future) or NFS 0 TFTP (future) 1 NFS Data[10, 11] Threshold</p>
Major	0x43	DSP Series 2 File Space Out of Memory	0x0C	<p>Data[0-4] Slot AIB (0x01) Data[5, 6] Number of Files in Cache Data[7, 8] Number of Free Blocks Needed Data[9] TFTP (future) or NFS 0 TFTP (future) 1 NFS Data[10, 11] Threshold</p>
Minor	0x44*	NFS Traffic Minor Threshold Alarm	0x10	<p>Data[0-4] Slot AIB (0x01) Data[5] Alarm Type (See list below) 0 (0x00) Number of Reads 1 (0x01) Total Bytes Read 2 (0x02) Average Read Delay 3 (0x03) Maximum Read Delay 4 (0x04) Number of Writes 5 (0x05) Total Bytes Written 6 (0x06) Average Write Delay 7 (0x07) Maximum Write Delay 8 (0x08) Maximum Simultaneous Files Opened 9 (0x09) Average Simultaneous Files Opened 10 (0x0A) CPU Idle Time 11 (0x0B) VRA Process Delay 12 (0x0C) VRA IO Queue Delay Data[6-9] Value of Statistics (Units vary) Data[10] M Value (Hits in window) Data[11] N Value (Samples in window) (N>=M) Data[12-15] Threshold value (if this is a time value, it is in microseconds.)</p> <p>The M-value indicates number of times the threshold is exceeded in the sample window of size N. See the <i>CSP Developers Guide Overview</i>, DSP2 Overload Management for more data.</p>

Alarm 0x00B9

Major	0x45*	NFS Traffic Threshold Major Alarm	0x10	<p>Data[0-4] Slot AIB (0x01)</p> <p>Data[5] Alarm Type (See list below.)</p> <p>0 (0x00) Number of Reads</p> <p>1 (0x01) Total Bytes Read</p> <p>2 (0x02) Average Read Delay</p> <p>3 (0x03) Maximum Read Delay</p> <p>4 (0x04) Number of Writes</p> <p>5 (0x05) Total Bytes Written</p> <p>6 (0x06) Average Write Delay</p> <p>7 (0x07) Maximum Write Delay</p> <p>8 (0x08) Maximum Simultaneous Files Opened</p> <p>9 (0x09) Average Simultaneous Files Opened</p> <p>10 (0x0A) CPU Idle Time</p> <p>11 (0x0B) VRA Process Delay</p> <p>12 (0x0C) VRA IO Queue Delay</p> <p>Data[6-9] Value of Statistics (Units vary)</p> <p>Data[10] M Value (Hits in window)</p> <p>Data[11] N Value (Samples in window) (N>=M)</p> <p>Data[12-15] Threshold value (if this is a time value, it is in microseconds.)</p> <p>The M-value indicates number of times the threshold is exceeded in the sample window of size N. See <i>CSP Developers Guide Overview</i>, <i>DSP2 Overload Logic Overload Management</i> for more data.</p>
Major	0x46	DSP Series 2 TFTP Communications Error	0x07	<p>Data[0-4] Slot AIB (0x01)</p> <p>Data[5, 6] Error ID</p>
Major	0x47	NFS Read Error	0x0B	<p>Data[0-4] Slot AIB (0x01)</p> <p>Data[5, 6] Server ID</p> <p>Data[7-10] File ID</p> <p>Data[11-14] Bytes written to server up to time of failure</p>
Major	0x48	NFS Write Error	0x0B	<p>Data[0-4] Slot AIB (0x01)</p> <p>Data[5, 6] Server ID</p> <p>Data[7-10] File ID</p> <p>Data[11-14] Bytes written to server up to time of failure</p>
Inform.	0x49	Vocabulary Index File Read	0x0F	<p>Data[0-4] Slot AIB (0x01)</p> <p>Data[5, 6] Server ID</p> <p>Data[7-10] Checksum</p> <p>Data[11-14] File ID (Last successfully loaded VIF line)</p>
Inform.	0x4A	NFS Server Unmounted	0x07	<p>Data[0-4] Slot AIB (0x01)</p> <p>Data[5, 6] Server ID</p>
Major	0x4B	NFS Server Unmount Error	0x07	<p>Data[0-4] Slot AIB (0x01)</p> <p>Data[5, 6] Server ID</p>
Major	0x4C	NFS Open File Error	0x0B	<p>Data[0-4] Slot AIB (0x01)</p> <p>Data[5] Server ID (MSB)</p> <p>Data[6] Server ID (LSB)</p> <p>Data[7] File ID (MSB)</p> <p>Data[8] File ID . . .</p> <p>Data[9] File ID . . .</p> <p>Data[10] File ID (LSB)</p>

Alarm 0x00B9

Major	0x4D	File Queue Failed	0x0D	<p>Data [0-4] Slot AIB Data[5] Parent Conference ID MSB Data[6] Parent Conference ID LSB Data[7] Child Conference ID MSB Data[8] Child Conference ID LSB Data[9] File ID (MSB) Data[10] File ID ... Data[11] File ID ... Data[12] File ID (LSB)</p> <hr/> <p>In the case of a parent conference, the Child Conference ID bits (Data[7-8]) shall be 0xFFFF.</p>
Inform.	0x4E	Error reading a .wav file	0x0E	<p>Data [0-4] Slot AIB Data [5-6] Server ID Data [7-10] File ID Data [11] Alarm Subtype (see below) Data [12-15] Value (Incorrect value found in file, see below) Subtype - Value (0xn timer)</p> <ul style="list-style-type: none"> 0x00 - No RIFF Chunk Found - Name of first chunk 0x01 - RIFF type .wav not found - RIFF type found 0x02 - No Format subchunk found - N/A (0x00000000) 0x03 - Invalid number of channels - Number of channels 0x04 - Invalid number of samples per second - Number of samples per second 0x05 - Invalid number of bits per second - Number of bits per second 0x06 - Unsupported encoding format - Encoding Format 0x07 - Data subchunk not found - N/A (0x00000000)

Alarm 0x00B9

Major	0x4F	Ethernet Switchover Occurred		Data [0-4] Slot AIB Data [5] Bit mask Bit 0 Interface B link state: 0=Down. 1=Up Bit 1 Interface A link state: 0=Down. 1=Up Bit 2 The active Ethernet Interface: 0=Interface A 1= Interface B Diagnose the LAN connection between the DSP2 card and the NFS server.
Inform.	0x4F	Ethernet Switchover Occurred (DSP Series 2 card)	0x06	Data [0-4] Slot AIB Data [5] Active Port: 01, 02, or 03 <u>Port</u> <u>Interface</u> 01 NET1 02 NET2 03 NET3
Major	0x50	DSP Ethernet Unavailable	0x04	Ethernet address of the DSP Series 2 Card. If at any point the RCOMM link fails, this alarm is sent to the host.
Major	0x52	ISDN Ethernet Unavailable	0x04	Ethernet address of the ISDN Series 3 Card. If at any point the RCOMM link fails, this alarm is sent to the host.
Inform.	0x58	Bad message received from local host	0x06	0x01 Slot AIB Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x01) Data[3] Data Length (0x01) Data[4] Slot Number Data[5] Error Reason 0x00 Unknown cause of error message 0x01 Frame symbol (0xFE) caused error message 0x02 Escape symbol (0xFD) error 0x03 Message has zero length 0x04 Message has bad checksum 0x05 Message has bad length
Minor	0x59	Vocabulary Index File Read Incomplete	0x13	Data[0-4] Slot AIB (0x01) Data[5, 6] Server ID Data[7-10] Checksum Data[11-14] File ID (Last successfully loaded VIF line) Data[15-18] VIF bytes loaded into memory successfully
Inform.	0x5A	Vocabulary Index File Potential Delay	0x0B	Data[0-4] Slot AIB (0x01) Data[5, 6] Server ID

*Followed by an *Alarm Cleared* message

0x03 Span Alarms

Severity	Alarm (Number/Name)	Data Length	Data/Troubleshooting
Major	0x01 Carrier Group Alarm (CGA) or Span Dead (Please refer to the <i>Span Filter Configure</i> message)	0x07	<p>0x0C Logical Span AIB</p> <p>Span Status (a bit mask that indicates what is being sent and received on the span):</p> <ul style="list-style-type: none"> Bit 0 0x01 Receiving Red Bit 1 0x02 Receiving Remote Alarm Indication/Yellow Bit 2 0x04 Receiving Loss of Signal Bit 3 0x08 Receiving Out of Frame Bit 4 0x10 Sending Red (AIS) Bit 5 0x20 Sending Remote Alarm Indication/Yellow Bit 6 0x40 Receiving AIS Bit 7 0x80 Receiving constant E-bit with Remote Alarm Indication (valid for E1 with insert CRC and Euro-ISDN) <hr/> <p>This alarm is generated when the CSP detects a carrier group alarm for more than the configured time period.</p> <p>Either you lost a clock source or someone pulled the patch cord and broke the physical connection to the TDM (Time Division Multiplexed) network.</p> <p>Request operational personnel find out why the physical connection has been compromised between the CSP to the telecom network. Also look at the way the span has been configured - it must be the same on both ends. See the <i>T1 Span Configure</i>, <i>E1 Span Configure</i> or <i>J1 Span Configure</i> messages for different configuration parameters.</p>
	0x02 Reserved		
Major	0x03 Excessive Burst of Slips	0x08	<p>0x0C Logical Span AIB</p> <p>Slot Number</p> <p>Span Offset</p> <hr/> <p>This notification occurs when the CSP detects an abnormally high number of slips. The address information represents the logical span of the span.</p> <p>There might be noisy T1 signals on the line, a configuration mismatch that is, B8Zs/AMI/SF/ESF etc. for T1, HDB3, CRC-4 etc for E1 spans or a bad timing source for span synchronization.</p> <p>Use the Span Status Query message (0x0084) which will return all Layer 1 parameters. See the <i>API Reference</i> for more details.</p> <p>Also, you can check your timing source. Most of the time the CSP uses a Span from the network to Time the entire CSP chassis. If that source is not good, it may trigger slips and carrier Group Alarms (CGA) on other spans.</p>
Major	0x04 J1-Specific Failure	0x06	0x0C Logical Span AIB

Major	0x05	Logical Span ID already exists on another node	0x06	0x0C Logical Span AIB An application is trying to configure a Logical Span ID that already exists on another node. There could be a software error in SwitchMgr or an error in the configuration file. Check the configuration. Collect the alarm.log and socket.log and contact Dialogic Technical Support.
Inform.	0x0F	CGA Filter set to 0xFFFF	0x06	0x0C Logical Span AIB

*Followed by an *Alarm Cleared* message

0x04 Channel Alarms

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
-	0x01	Reserved		
Inform.	0x02	ISDN D Channel Establishing (the distant end is not communicating on the D Channel)	0x07	0x0D Channel AIB ISDN D channel is trying to get established and the distant end is not communicating on the D channel. Initial configuration or the distant end has stopped communicating on the D channel. Investigate the distant end D channel.
Inform.	0x03	PPL Error Data	Variable	Refer to the PPL Error Data at the end of this message for the data format.
	0x04	Switchover Initiate		Check D Channel Out of Service condition. The Primary D channel has gone from being in-service to out-of-service. The address information represents the slot and channel of the D channel. Look for Card or Span alarms that would also indicate a hardware failure. Reset the D Channel if it does not come back in service contact remote end network administrator to check remote end status.
	0x05	Switchover Success		The D channel switchover was successful. The new D channel is now the controller. The address information represents the span and channel of the new D channel. Investigate the cause of the switchover.

	0x06	Switchover Failure		<p>The controller switchover failed. The existing D channel is still the controlling D channel. The address information represents the span and channel of the new D channel.</p> <p>Investigate cause of switchover and state of the Backup D Channel.</p>
	0x07	Backup Diagnostic		<p>Information Alarm when the state of the D channel changes. The address information represents the span and channel of the D channel.</p> <p>Investigate the cause of D Channel switchover.</p>
	0x08	SS IPDC Error		<p>Look for other error messages that would indicate a network or system condition leading to this problem. If symptoms persist, contact Dialogic Technical Support.</p>
Major	0x09	Invalid Associated Channel	0x09	<p>The Device Server ID and 0x0D Channel AIB.</p>
Major	0x0A	DSP Resource Wait Timeout	0x09	<p>The Device Server ID and 0x0D Channel AIB.</p> <p>A DSP Resource Wait timeout occurs when the CSP gets no response when requesting a physical channel.</p> <p>Could be a hardware failure. If the problem persists, contact Dialogic Technical Support</p>
Major	0x0E	EXNET Connection Error		<p>Contact Dialogic Technical Support.</p>
Inform.	0x11	DSP Service Cancel Timeout Severity	0x09	<p>Data[0-6] 0x0D Channel [7-8] L4 CH State</p>

*Followed by an *Alarm Cleared* message

0x05 DSP SIMM Alarms

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Inform.	0x01	DSP SIMM Taken Out of Service	0x06	<p>0x12 DSP SIMM AIB.</p> <p>Check with operations personnel to verify card removal.</p> <p>Verify the host did not send a <i>Service State Configure</i> (0x000A) message to take the SIMM out of service. Also verify that the DSP card is still installed and powered on.</p>
-	0x02	Reserved	0x00	No Data

Alarm 0x00B9

Inform.	0x03	Recorded Announcement Erase Complete	0x00	No Data Sent upon completion of erasure. When you have erased a VRA SIMM using <i>Recorded Announcement</i> (0x0054), this alarm indicates that the erasure is complete. The host must wait for this alarm before it begins any download.
---------	------	--------------------------------------	------	--

Alarm 0x00B9

Major	0x04	Recorded Announcement Download Inconsistency	0x00	<p>No Data</p> <p>Sent upon completion of download. This alarm indicates an inconsistency between the information in a <i>Recorded Announcement Download 0x0052</i> message and the actual announcement in the <i>Recorded Announcement Download 0x0053</i> message. The inconsistency is usually caused by an incorrect checksum or announcement size. The announcement information is stored as it appears in the <i>Recorded Announcement Download 0x0053</i> message. Play out the announcement to check it.</p> <p>Could be caused by an incorrect checksum or announcement size. Check the announcement and delete one of the announcements.</p>
Major	0x05	Recorded Announcement ID Inconsistency (on same card)	0x00	<p>No Data</p> <p>Sent when a VRA SIMM is brought in service. This alarm indicates that an announcement uses the same ID as another announcement, with different information, on the DSP card. Both announcements have the same ID, and either announcement might be played when the ID is specified in the <i>Recorded Announcement Connect 0x0055</i> message.</p> <p>Indicates that the host downloaded an announcement using the same ID as another announcement, with different information on the DSP card.</p> <p>Check the configuration to ensure that you are not using the same ID for different RAN files.</p>
Inform.	0x06	Recorded Announcement Download Ready	0x08	<p>An AIB indicating the slot number:</p> <p>Data[0] Address Method (0x00)</p> <p>Data[1] No. of Address Elements (0x01)</p> <p>Data[2] Address Type (0x12)</p> <p>Data[3] Data Length (0x02)</p> <p>Data[4] Slot Number</p> <p>Data[5] SIMM Number</p> <p>Data[6, 7] RAN ID</p> <hr/> <p>Sent upon completion of download. The DSP-ONE card will not accept another Recorded Announcement Download Initiate message until it has sent this alarm.</p>

Inform	0x07	Recorded Announcement File System Conversion Success	0x06	<p>Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x12) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] SIMM Number</p> <hr/> <p>Sent after the flash memory file system is converted. Informs the host that the RANs are now stored in the requested file system.</p> <p>This alarm takes three to five minutes to be generated after the host sends the file system conversion request.</p> <p>After this alarm you are able to bring the VRAS simm into service.</p>
Major	0x08	Recorded Announcement File System Conversion Failure	0x07	<p>Data Length: (0x07) Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x12) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] SIMM Number Data[6]:Failure Code</p> <hr/> <p>Failure codes: VRA_FAILURE = 0, Amount of Flash Data invalid for New File System.</p> <p>VRA_FLASH_WRITE_FAILURE = 2, Writing to Flash failed.</p> <p>VRA_FLASH_ERASE_FAILURE = 3, Erasing Flash failed.</p> <p>VRA_RAM_WRITE_FAILURE = 4, Writing RAM failed.</p> <p>VRA_DATA_VALIDATION_FAILURE = 5 CRC validation failed.</p> <p>Sent after an unsuccessful attempt to convert the file system. Informs the host that the file system conversion process failed. You should delete the RANs on this VRA SIMM.</p> <p>If the file system convert fails then delete all RANs from this SIMM and try again.</p>

Alarm 0x00B9

Major	0x09	Recorded Announcement File System Timeout	0x06	<p>Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x12) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] SIMM Number</p> <hr/> <p>Sent after an attempt to convert the file system times out, if the attempted deletion of entire flash memory, file system conversion, or defragmentation fails to complete in the time allowed.</p> <p>Try deleting each RAN from the SIMM.</p>
Inform	0x0B	Recorded Announcement Defragmentation Success	0x06	<p>Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x12) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] SIMM Number</p> <hr/> <p>Sent after defragmentation of the flash memory is complete. Informs the host that the RANs now take up as little flash memory as possible.</p> <p>This alarm takes three to five minutes to be generated after the host sends the file system Defragment request.</p>
Major	0x0C	Recorded Announcement Defragmentation Failure	0x07	<p>Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x12) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] SIMM Number Data[6] Failure Code (see failure codes listed under Alarm File System Conversion Failure 0x08)</p> <hr/> <p>Sent after an unsuccessful attempt to defragment flash memory. Informs the host that the flash memory defragmentation has failed. The host should delete the flash memory on this VRA SIMM.</p>

Alarm 0x00B9

Inform	0x0D	Recorded Announcement Single Deletion Complete	0x08	<p>Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x12) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] SIMM Number Data[6] RAN ID (MSB) Data[7] RAN ID (LSB)</p> <hr/> <p>Sent in response to the message, <i>Recorded Announcement Single Delete</i>. Informs the host that the RAN ID has been marked for deletion and is no longer available for use.</p> <p>This alarm takes zero to 30 seconds to be generated after the host sends the delete request.</p>
Inform	0x0E	Recorded Announcement Need Defragmentation	0x07	<p>Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x12) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] SIMM Number Data[6] Defragmentation Level</p> <hr/> <p>Sent after a Recorded Announcement Single Delete has been processed and the flash memory where the RANs are stored has reached the highest level of fragmentation. Indicates that enough of the flash memory is occupied by deleted RANs to make it worthwhile to defragment the flash memory.</p>

*Followed by an *Alarm Cleared* message

0x06 DSP Alarms

Severity	Alarm (Number/Name)	Data Length	Data/Troubleshooting
Major	0x01 DSP Out of Service	0x03	Data[0] Slot Number Data[1] Module Number (0-1) Data[2] Chip Number (0-3)
			<p>Either the user took the DSP chip out of service using a service state configuration message addressing the chip or the DSP mother card took the DSP chip out of service due to lose of polls from the DSP mother card.</p> <p>If SwitchMgr does not bring the DSP chip back into service using the service state configuration message addressing the DSP chip, then collect the alarm.log and socket.log and call Dialogic Technical Support.</p>
Inform.	0x02* Resources Exceed Limit	0x07	Data[0] Function Type Data[1] Alarm Threshold Data[2] Alarm Cleared Threshold Data[3, 4] Number of Channels Used Data[5, 6] Number of Channels Configured for Function Type Data[7,8] Clear threshold for this function type
Major	0x03 DSP Playback Underrun	0x08	Data[0] Slot Number Data[1] Module Number (0-1) Data[2] Chip Number (0-3) Data[3] TFTP (future) or NFS Data[4,5] Total temporary storage Blocks Data[6,7] Number of Blocks in use by DSP
Major	0x04 DSP Record Overrun	0x07	Data[0] Slot Number Data[1] Module Number (0-1) Data[2] Chip Number (0-3) Data[3-6] File ID
Major	0x05 DSP Temporary Storage Minor Alarm	0x0D	Data[0] Slot Number Data[1] Module Number (0-1) Data[2] Chip Number (0-3) Data[3-6] File ID Data[7,8] Total Number of Blocks Data[9,10] Number of Blocks on DSP Data[11,12] Threshold

Alarm 0x00B9

Major	0x06 DSP Temporary Storage Full	0x0D	Data[0] Data[1] Data[2] Data[3-6] Data[7,8] Data[9,10] Data[11,12]	Slot Number Module Number (0-1) Chip Number (0-3) File ID Requested Number of Blocks Number of used Blocks on DSP Number of free Blocks on DSP
Inform.	0x07* DSP Points Exceed Limit	0x0A	Data[0] Data[1] Data[2-5] Data[6-9]	Alarm Threshold Alarm Clear Threshold Resource Points in use Resource Points Configured
Major	0x08* DSP Chip Reset	0x03	Data[0] Data[1] Data[2]	Slot Module Number Chip Number

*Followed by an *Alarm Cleared* message

0x07 Node Alarms

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Minor	0x01	Not Using Extended API	0x06	Data[0] Physical Node ID, MSB Data[1] Physical Node ID Data[2] Physical Node ID Data[3] Physical Node ID, LSB Data[4] Logical Node ID Data[5] Host Node ID
Inform.	0x02	Available Conferencing Timeslot Changed	0x02	Data[0] New Timeslot Count, MSB Data[1] New Timeslot Count, LSB
Inform.	0x05	Unable to Communicate with Node		Normally when this alarm is sent to the host, it includes the filename and line number where the error occurred. Send this information to Dialogic Technical Support.
Major	0x06	Remote SS7 Server Node Unavailable		Data[0] Node ID Data[1] Board Slot Number Check with network operations for other network alarms that could be related.
	0x07	0x07 Remote SS7 CIC Matrix Node Unavailable		Check with network operations for other network alarms that could be related.
Major	0x08	Remote SS7 Card Unavailable		Data[0] Number of IDs (always 1) Data[1] 1=Active SS7 Card 2=Standby SS7 Card Ensure that the Ethernet cable is plugged in CCS I/O Ethernet Port A. Check for faulty hardware, (SS7PQ, SS7HP, CCS I/O, ethernet cable, and Ethernet switch). Check with network operations for other network alarms that could be related.
Major	0x0A	Maximum Node Count Reached		
Major	0x0B	H.100 Bus Failure		This alarm occurs if the H.100 bus fails. For example, if there are no clock signals to drive the H.100 bus.
Major	0x0C	Clock B Failure		This alarm occurs if the clock B signal of the H.100 bus fails. This alarm notifies the host about the clock signal failure, so that the host can appoint another board to source the clock or take other actions.

Alarm 0x00B9

Major	0x0D	Clock A Failure		This alarm occurs if the clock A signal of the H.100 bus fails. This alarm notifies the host about the clock signal failure, so that the host can appoint another board to source the clock or take other actions.
Major	0x0E	Cannot Be Primary A		This alarm occurs when the board could not become the primary master driving clock A, when instructed to do so. This alarm is sent to the host to enable it to appoint another board as primary master driving clock A or take other actions.
Major	0x0F	Cannot Be Primary B		This alarm occurs when the board could not become the primary master driving clock B, when instructed to do so. This alarm is sent to the host to enable it to appoint another board as primary master driving clock B or take other actions.
Major	0x10	Cannot Be Secondary A		This alarm occurs when the board could not become the secondary master driving clock A, when instructed to do so. This alarm is sent to the host to enable it to appoint another board as secondary master driving clock A or take other actions.
Major	0x11	Cannot Be Secondary B		This alarm occurs when the board could not become the secondary master driving clock B, when instructed to do so. This alarm is sent to the host to enable it to appoint another board as secondary master driving clock B or take other actions.
Major	0x12	Momentary Failure of Clock A		This alarm occurs when a momentary failure of clock A is observed and clock A recovers.
Major	0x13	Momentary Failure of Clock B		This alarm occurs when a momentary failure of clock B is observed and clock B recovers.
Major	0x14	Taking Over As Primary A		This alarm occurs when the board is acting as the secondary master, driving clock A and the primary master, driving clock B fails. Since the primary master, driving clock B has failed, the secondary master driving clock A takes over as the primary master, driving clock A. The board is taking over as the primary master driving clock A.

Alarm 0x00B9

Major	0x15	Taking Over As Primary B		This alarm occurs when the board is acting as the secondary master, driving clock B and the primary master, driving clock A fails. Since the primary master, driving clock A has failed, the secondary master driving clock B takes over as the primary master, driving clock B. The board is taking over as the primary master driving clock B.
-------	------	--------------------------	--	--

*Followed by an *Alarm Cleared* message

0x08 DS3 Alarms

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Major	0x01	DS3 Alarm	0x07	Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x32) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] DS3 Offset Data[6] Alarm (a bit mask that indicates what is being sent and received on the DS3): Bit 0 0x01 Receiving Red Bit 1 0x02 Receiving Remote Alarm Ind. / Yellow Bit 2 0x04 Receiving Loss of Signal Bit 3 0x08 Receiving Out of Frame Bit 4 0x10 Receiving AIS Bit 5 0x20 Sending AIS Bit 6 0x40 Sending Remote Alarm Ind. / Yellow Bit 7 Reserved <hr/> Check for clock reference alarms that could be associated with this failure.

Alarm 0x00B9

Inform.	0x02	Far End Alarm and Control (FEAC)	<p>0x01 If Severity is Informative</p> <p>0x07</p>	<p>Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x32) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] DS3 Offset Data[6] Alarm</p> <ul style="list-style-type: none"> 0x00 Activate Span Offset 0 in Loopback Mode 0x01 Activate Span Offset 1 in Loopback Mode 0x02 Activate Span Offset 2 in Loopback Mode 0x03 Activate Span Offset 3 in Loopback Mode 0x04 Activate Span Offset 4 in Loopback Mode 0x05 Activate Span Offset 5 in Loopback Mode 0x06 Activate Span Offset 6 in Loopback Mode 0x07 Activate Span Offset 7 in Loopback Mode 0x08 Activate Span Offset 8 in Loopback Mode 0x09 Activate Span Offset 9 in Loopback Mode 0x0A Activate Span Offset 10 in Loopback Mode 0x0B Activate Span Offset 11 in Loopback Mode 0x0C Activate Span Offset 12 in Loopback Mode 0x0D Activate Span Offset 13 in Loopback Mode 0x0E Activate Span Offset 14 in Loopback Mode 0x0F Activate Span Offset 15 in Loopback Mode 0x10 Activate Span Offset 16 in Loopback Mode 0x11 Activate Span Offset 17 in Loopback Mode 0x12 Activate Span Offset 18 in Loopback Mode 0x13 Activate Span Offset 19 in Loopback Mode 0x14 Activate Span Offset 20 in Loopback Mode 0x15 Activate Span Offset 21 in Loopback Mode 0x16 Activate Span Offset 22 in Loopback Mode 0x17 Activate Span Offset 23 in Loopback Mode 0x18 Activate Span Offset 24 in Loopback Mode 0x19 Activate Span Offset 25 in Loopback Mode 0x1A Activate Span Offset 26 in Loopback Mode 0x1B Activate Span Offset 27 in Loopback Mode 0x1C Activate All Span Offsets in Loopback Mode 0x1D Activate DS3 in Loopback Mode
---------	------	----------------------------------	--	--

				<p>0x20 Deactivate Span Offset 0 in Loopback Mode 0x21 Deactivate Span Offset 1 in Loopback Mode 0x22 Deactivate Span Offset 2 in Loopback Mode 0x23 Deactivate Span Offset 3 in Loopback Mode 0x24 Deactivate Span Offset 4 in Loopback Mode 0x25 Deactivate Span Offset 5 in Loopback Mode 0x26 Deactivate Span Offset 6 in Loopback Mode 0x27 Deactivate Span Offset 7 in Loopback Mode 0x28 Deactivate Span Offset 8 in Loopback Mode 0x29 Deactivate Span Offset 9 in Loopback Mode 0x2A Deactivate Span Offset 10 in Loopback Mode 0x2B Deactivate Span Offset 11 in Loopback Mode 0x2C Deactivate Span Offset 12 in Loopback Mode 0x2D Deactivate Span Offset 13 in Loopback Mode 0x2E Deactivate Span Offset 14 in Loopback Mode 0x2F Deactivate Span Offset 15 in Loopback Mode 0x30 Deactivate Span Offset 16 in Loopback Mode 0x31 Deactivate Span Offset 17 in Loopback Mode 0x32 Deactivate Span Offset 18 in Loopback Mode 0x33 Deactivate Span Offset 19 in Loopback Mode 0x34 Deactivate Span Offset 20 in Loopback Mode 0x35 Deactivate Span Offset 21 in Loopback Mode 0x36 Deactivate Span Offset 22 in Loopback Mode 0x37 Deactivate Span Offset 23 in Loopback Mode 0x38 Deactivate Span Offset 24 in Loopback Mode 0x39 Deactivate Span Offset 25 in Loopback Mode 0x3A Deactivate Span Offset 26 in Loopback Mode 0x3B Deactivate Span Offset 27 in Loopback Mode 0x3C Deactivate All Span Offsets in Loopback Mode 0x3D Deactivate DS3 in Loopback Mode</p> <hr/> <p>If a major alarm is indicated, check network operations for far-end CGA or other facility alarms.</p> <p>If the severity is informative, you can enable loopback mode or ignore.</p> <p>If the severity is major, sync with the carrier and verify Layer 1 integrity</p>
--	--	--	--	--

Alarm 0x00B9

Major	0x02	Far End Alarm and Control (FEAC)	0x03 If Severity is Major 0x07	Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x32) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] DS3 Offset Data[6] Alarm 0x01 DS3 Equipment Failure (Affecting Service) 0x02 DS3 Loss of Signal 0x03 DS3 Out of Frame 0x04 DS3 AIS 0x05 DS3 Idle 0x06 DS3 Equipment Failure (Not Affecting Service) 0x07 Common Equipment Failure 0x08 DS1 Multiple Loss of Signal 0x09 DS1 Equipment Failure (Affecting Service) 0x0A DS1 Single Loss of Service 0x0B DS1 Equipment Failure (Not Affecting Service)
Major	0x03*	DS3 Application Mismatch	0x07	Alarm is send when the received DS3 Application bit does not match the configured DS3 framer mode. Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x32) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] DS3 Offset Data[6] Application ID bit received 0x00 - M13 0x01 - C-Bit

*Followed by an *Alarm Cleared* message

0x09 VoIP Module Alarms

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Major	0x00	Module In Service	0x06	<p>See <i>0x42 VoIP Module (2-22)</i></p> <p>A VDAC-ONE or IPN2 module has come in service. The address information represents the slot and VDAC module ID.</p> <hr/> <p>No action required.</p>
Major	0x01	Module Out of Service	0x06	<p>See <i>0x42 VoIP Module (2-22)</i></p> <hr/> <p>The VDAC or IPN2 module is out-of-service. The address information represents the slot and VDAC-ONE or IPN2 module ID.</p> <p>Verify the module is no longer needed for the operation.</p>
Major	0x02	Module Dead	0x06	<p>See <i>0x42 VoIP Module (2-22)</i></p> <hr/> <p>The VDAC-ONE or IPN-2 module is no longer communicating. The address information represents the slot and VDAC-ONE/IPN-2 module ID.</p> <p>Review Field Bulletin O-4083 at www.cantata.com/support</p> <p>If problem persists, contact Dialogic Technical Support</p>
Major	0x03	DSP Dead	0x07	<p>Data[0] Address Method Data[1] Number of Address Elements Data[2] Address Type (0x42) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] Module Number Data[6] DSP Number 0x00-0x07</p> <hr/> <p>The DSP on VDAC-ONE or IPN-2 card is no longer communicating.</p> <p>The address information represents the slot and VDAC/IPN2 module ID.</p> <p>Check the IP network for the source of the error. If the problem persists, contact Dialogic Technical Support.</p>

Alarm 0x00B9

Major	0x04	Module Reset	0x06	<p>See <i>0x42 VoIP Module (2-22)</i></p> <p>Check with the operations personnel to verify board reset.</p>
Major	0x05	Module Resetting	0x06	<p>See <i>0x42 VoIP Module (2-22)</i></p> <hr/> <p>A VDAC-ONE or IPN2 module is currently resetting. The address information represents the slot and VDAC-ONE or IPN2 module ID.</p> <p>Software fault could have occurred or the VoIP card or the module is being reset by the host.</p> <p>If the host did not issue the reset request, then perform the Fault Log Query for the VoIP module indicated in the message and contact Dialogic Technical Support.</p>
Major	0x06	DSP Reset	0x07	<p>Data[0] Address Method Data[1] Number of Address Elements Data[2] Address Type (0x42) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] Module Number Data[6] DSP Number 0x00-0x07</p> <hr/> <p>This alarm is sent to the host upon reset of a DSP Resource on a VDAC-ONE card's VoIP Module or IPN-2 card's VoIP Module.</p> <p>Too many fatal errors are detected on this DSP module.</p> <p>Check the IP network for the source of the error. If the problem persists, contact Dialogic Technical Support.</p>
Major	0x07	VMOD Inaccessible	0x06	<p>Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x42) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] Module Number</p> <hr/> <p>This alarm is sent to the host when a module becomes inaccessible from the motherboard.</p> <p>See also <i>0x42 VoIP Module (2-22)</i></p>

Alarm 0x00B9

Major	0x08	Fatal Calisto Error	0x07	<p>Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x42) Data[3] Data Length (0x02) Data[4] Module Number Data[5] Calisto Number</p> <hr/> <p>This alarm results from the host sending the Media Recovery Method TLV (0x01F4) in the <i>Resources Attribute Configure</i> (0x00E3) message.</p>
Inf.	0x09	Media Recovery Procedure Started		<p>Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x42) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] Module Number Data[6] Calisto Number - set to 0xFF in case of Immediate forced bleed-off.</p> <hr/> <p>This alarm results from the host sending the Media Recovery Method TLV (0x01F4) in the <i>Resources Attribute Configure</i> (0x00E3) message.</p>
Inf.	0x17	Software Error	Variable	<p>See <i>0x42 VoIP Module (2-22)</i> Data[0] Address Method (0x00) Data[1] Number of Address Elements (0x01) Data[2] Address Type (0x42) Data[3] Data Length (0x02) Data[4] Slot Number Data[5] Module Number Data[6-45] Filename (in ASCII) Data[46,47] Line Number Data[48-n] Diagnostic Data</p>

*Followed by an *Alarm Cleared* message

0x1E TFTP Alarm

NOTE: This alarm is resent every 20 seconds as long as the alarm condition remains

Length Variable

Data[0] Alarm Code

- 0x00 Cannot TFTP file
- 0x01 Invalid TFTP Configuration File Format
- 0x02 Invalid BRecord Format
- 0x03 RAM Corrupted
- 0x04 Old Timestamp
- 0x05 No timestamp supplied

Data[1] Type

- 0x02 Alarm Status (See Data[2], below)
- All other types: Dialogic Use Only. Report to Dialogic Technical Support.

Data[2] Status

Applies to Alarm Type 0x02 (Alarm Status) only.

- 0x00 No Status
- 0x01 TFTP device detected a protocol error
(such as a non-Data packet received or expected ACK not received. Occurs when a file is not found).
- 0x02 TFTP device timed out (Incorrect IP Address)
- 0x03 TFTP device is out of sync
- 0x04 TFTP device has no more free socket IDs
(Occurs when another TFTP Download was in progress but was aborted)
- 0x05 TFTP device cannot use a channel number given to it
- 0x06 TFTP device has not been initialized
- 0x07 No server IP Address has been given
- 0x08 No TFTP Configuration filename has been given
- 0x09 No timestamp has been given
- 0x0A A TFTP Configuration or BRecord File line is too long
- 0x0B Too much data for internal buffer
- 0x0C TFTP Configuration File line has no '='
- 0x0D TFTP Configuration File line has no second '='
- 0x0E TFTP Configuration File timestamp is missing a parameter
- 0x0F Load filename is greater than 127 characters
- 0x10 EX/CPU Label in TFTP Configuration file not set to SAVE_LOAD_TRUE
- 0x11 Not enough RAM on EX/CPU card to save all loads that are configured to be saved
- 0x12 BRecord File is out of sync. There is no 'B' starting the record.
- 0x13 BRecord receiver is out of sync. Expected a different record type.
- 0x14 Invalid BRecord Checksum
- 0x15 Invalid BRecord Count
- 0x16 Incorrect Number of Bytes in BRecord Load

Alarm 0x00B9

- 0x17 Invalid Load Checksum
- 0x18 Incorrect Load for Card
- 0x19 Load is from a different software version than the version on the EX/CPU
- 0x1A Timeout waiting for response from internal download process
- 0x1B Host download is in progress
- 0x1C TFTP Download aborted by initiation of host download
- 0x1D Timeout waiting for buffer
- 0x1E Stopped by host starting another TFTP
- 0x1F Configuration file has a new save option for an existing load
- 0x20 Operating system general error
- 0x21 Operating system I/O error

Data[3] Load Number

- 0x00 ST1LC
- 0x01 EX/CPU
- 0x02 MFDSP
- 0x03 Reserved
- 0x04 SE1LC
- 0x05 SJ1LC
- 0x06 SS7
- 0x07 ISDN PRI, DASS2/DPNSS
- 0x09 DSP-ONE
- 0x0B T-ONE, E-ONE
- 0xFF No Load

Data[4] Error Information

- 0x00 No Error
- 0x01 Battery-backed configuration validation
- 0x02 System data validation
- 0x03 Input parameter validation
- 0x04 TFTP device initialization
- 0x05 TFTP device open
- 0x06 TFTP device read
- 0x07 TFTP device control
- 0x08 Loading Configuration File
- 0x09 Validating Configuration File
- 0x0A Loading Load Information
- 0x0B Validating Load Information
- 0x0C Sending "Begin" to download the state machine
- 0x0D Sending BRecord to download the state machine
- 0x0E Verifying TFTP completion
- 0x0F Sending "Complete" to download the state machine
- 0x10 Sending data buffer to line card
- 0x11 Verifying data for line card completion
- 0x21 Host Connection Dropped (Major) (Indicates that a connection to a host has been dropped)

PPL Error Data

The data format for this alarm depends on the PPL component
E1/T1 Components:

Length	0x 14
Data[0]	Component ID, MSB
Data[1]	Component ID, LSB
Data[2]	Logical Span ID
Data[3]	Channel
Data[4-10]	The Channel AIB
Data[4]	Address Method (0x00)
Data[5]	Number of Address Elements (0x01)
Data[6]	Address Type (0x0D)
Data[7]	Data Length (0x03)
Data[8]	Logical Span ID, MSB
Data[9]	Logical Span ID, LSB
Data[10]	Channel
Data[11]	PPL Protocol ID
Data[12]	Blocked
Data[13]	PPL Event, MSB
Data[12]	PPL Event, LSB
Data[14]	Initial State
Data[15]	Next State
Data[16]	Timestamp, MSB
Data[17]	Timestamp, LSB
Data[18]	Error Status, MSB
Data[19]	Error Status, LSB

See "PPL Audit Query 0xDD" for values of the various PPL bytes.

ISDN and DASS Components:

Length	0x 1A
Data[0]	Component ID, MSB
Data[1]	Component ID, LSB
Data[2]	Logical Span ID
Data[3]	Channel
Data[4-10]	The Channel AIB
Data[4]	Address Method (0x00)
Data[5]	Number of Address Elements (0x01)
Data[6]	Address Type (0x0D)
Data[7]	Data Length (0x03)
Data[8]	Logical Span ID, MSB

Data[9]	Logical Span ID, LSB
Data[10]	Channel
Data[11]	Subrate
Data[12]	Call Reference, MSB
Data[13]	Call Reference, LSB
Data[14]	Reserved
Data[15]	Reserved
Data[16]	PPL Protocol ID
Data[17]	Blocked
Data[18]	PPL Event, MSB
Data[19]	PPL Event, LSB
Data[20]	Initial State
Data[21]	Next State
Data[22]	Timestamp, MSB
Data[23]	Timestamp, LSB
Data[24]	Error Status, MSB
Data[25]	Error Status, LSB

See "PPL Audit Query 0xDD" for values of the various PPL bytes.

SS7 Components:

Length	Variable
Data[0]	Component ID, MSB
Data[1]	Component ID, LSB
Data[2-n]	See PPL Component Addressing for the AIB
Data[n]	PPL Protocol ID
Data[n]	Blocked
Data[n]	PPL Event, MSB
Data[n]	PPL Event, LSB
Data[n]	Initial State
Data[n]	Next State
Data[n]	Timestamp, MSB
Data[n]	Timestamp, LSB
Data[n]	Error Status, MSB
Data[n]	Error Status, LSB

VDAC-ONE and IP Network Interface Series 2 Component

Length	Variable
Data[0]	Component ID, MSB
Data[1]	Component ID, LSB
Data[2-n]	See PPL Component Addressing for the AIB
Data[n]	PPL Protocol ID
Data[n]	Blocked
Data[n]	PPL Event, MSB
Data[n]	PPL Event, LSB

Data[n] Initial State
 Data[n] Next State
 Data[n] Timestamp, MSB
 Data[n] Timestamp, LSB
 Data[n] Error Status, MSB
 Data[n] Error Status, LSB
 Data[n] External Initial State, MSB
 Data[n] External Initial State, LSB
 Data[n] External Next State, MSB
 Data[n] External Next State, LSB

0x0B SIP Alarm

Severity	Alarm (Number/Name)		Data Length	Data/Troubleshooting
Inform.	0x01	Received destination unreachable ICMP error for the SIP message sent.	0x16	Data[0-6] Channel AIB (0x0D) Data[7] ICMP Type (0x03) - Destination Unreachable Data[8] ICMP Code - For example 0x03 - Port Unreachable. Refer to RFC 792. Data[9] Protocol of failed packet (0x11) UDP Data[10-13] Source address of failed packet - the CSP SIP IP Address Data[14-17] Destination address of failed packet Data[18-19] Source port of failed packet Data[20-21] Destination port of failed packet
Inform.	0x02	Received invalid response for the SIP message sent	0x0A	Data[0-6] Channel AIB (0x0D) Data[7-8] Response Code Data[9] Error Code Error Codes: 0x00 Syntax Error 0x01 Transaction Not found 0x02 Invalid CSeq number 0x03 Tags mismatch 0x04 Invalid CSeq message type 0x05 Retransmitted response

Alarm Cleared 0x00C1

SwitchKit Name	AlarmCleared
Type	EXS API and SwitchKit API message
Description	<p>Alarm Cleared 0x00C1</p> <p>The CSP sends this message to indicate that an alarm previously reported in an <i>Alarm</i> message has been cleared because it has been resolved by a host action or by some external action or series of events.</p> <p>The following information about the originating alarm is reported:</p> <ul style="list-style-type: none"> • Severity • Alarm Type • Alarm Number • Alarm Data <p>See the <i>Alarm</i> message for Alarm Type, Alarm Number, and Data values. The <i>Data</i> will match that of the alarm that has been cleared, except for a span alarm, for which there is one byte of data indicating that the span status is 0x00 (Receiving Red) (0x000 for SwitchKit).</p>
Related Messages	Alarm 0x00B9
Sent by	CSP
SwitchKit Code	<p>C Structure</p> <pre>typedef struct { UBYTE InfoSize; UBYTE Severity; UBYTE Entity; UBYTE AlarmNum; UBYTE Info[249]; } XL_AlarmCleared;</pre> <p>C++ Class</p> <pre>class XLC_AlarmCleared : public XLC_InboundMessage { public: UBYTE getInfoSize() const; void setInfoSize(UBYTE x); UBYTE getSeverity() const; void setSeverity(UBYTE x); UBYTE getEntity() const; void setEntity(UBYTE x); UBYTE getAlarmNum() const; void setAlarmNum(UBYTE x); const UBYTE *getInfo() const; UBYTE *getInfo(); void setInfo(UBYTE *x); };</pre>

Resent for EXS API This message is sent once after five seconds.

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0005)
3, 4	Message Type (0x00C1)	3, 4	Message Type (0x00C1)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Severity	8	Checksum
9	Alarm Type		
10	Alarm Number		
11	Data Length (n)		
12	Data[0]		
:	Data[n]		
:	Checksum		

AllInService

Type	SwitchKit API message
Purpose	Use the <i>SK_AllInService</i> message to automatically bring into service all entities referenced in the configuration file.
Description	This message is for SwitchManager configuration files. It instructs SwitchManager to bring into service any entity referenced in configuration files using <i>ServiceStateConfig</i> . <i>AllInService</i> saves you from having to explicitly add <i>ServiceStateConfig</i> messages for each single entity.
Sent by	SwitchManager
Configuration	<i>AllInService</i> (Node = integer, ConnectionID = integer);

AllocateChannel

Type	SwitchKit API message
Description	Use the <i>SK_AllocateChannel</i> message to manually allocate a channel outside of the standard automatic allocation functions (for example, sk_watchChannelGroup() and sk_requestOutseizedChannel()). Regardless of the channel groups that the channel is in, the LLC tries to allocate the channel specified by Span and Channel to the calling application. On success, the return Status value contains 0x00. If the channel is already allocated to another application, Status contains SK_NO_CHANNELS.
Sent by	Application
C Structure	<pre>typedef struct { unsigned short Span; UBYTE Channel; } SK_AllocateChannel;</pre>
C Structure Response	<pre>typedef struct { int Status; } SK_AllocateChannelAck;</pre>
C++ Class	<pre>class SKC_AllocateChannel : public SKC_ToolkitMessage { public: unsigned short getSpan() const; void setSpan(unsigned short x); UBYTE getChannel() const; void setChannel(UBYTE x); };</pre>
C++ Class Response	<pre>class SKC_AllocateChannelAck : public SKC_ToolkitAck { public: int getStatus() const; void setStatus(int x); };</pre>

AllocateChannelGroup

Type	SwitchKit API message
Description	Use the <i>SK_AllocateChannelGroup</i> message to manually allocate all available channels in a given channel group. If <i>GroupName</i> is not valid, then <i>Status</i> in the acknowledgment contains <i>SK_INVALID_GROUP</i> . Otherwise, it contains 0x10, and <i>Channels</i> contains a list of all the channels that have been allocated. <i>Channels[0-1]</i> will contain the first span, <i>Channels[2]</i> will contain the first channel offset, and so on, for as many channels as are specified in <i>NumChannels</i> .
Sent by	Application
C Structure	<pre>typedef struct { char GroupName[50]; UBYTE reserved67[203]; } SK_AllocateChannelGroup;</pre>
C Structure Response	<pre>typedef struct { int Status; int NumChannels; UBYTE Channels[245]; } SK_AllocateChannelGroupAck;</pre>
C++ Class	<pre>class SKC_AllocateChannelGroup : public SKC_ToolkitMessage { public: const char *getGroupName() const; void setGroupName(const char *x); };</pre>
C++ Class Response	<pre>class SKC_AllocateChannelGroupAck : public SKC_ToolkitAck { public: int getStatus() const; void setStatus(int x); int getNumChannels() const; void setNumChannels(int x); const UBYTE *getChannels() const; UBYTE *getChannels(); void setChannels(UBYTE *x); };</pre>

AllOutOfService

Type	SwitchKit API message
Purpose	Use the <i>SK_AllOutOfService</i> message to automatically take all referenced entities out-of-service on any configured node prior to sending another configuration.
Description	<p>SwitchManager takes all entities found on any configured node out-of-service after receiving this message. Sending this message saves you from having to explicitly add <i>ServiceStateConfig</i> messages for each entity.</p> <p>Taking entities out-of-service prior to configuration is important to prevent system purges caused by configuring a channel that is in-service.</p> <p>Important! Use of <i>AllOutOfService</i> is not recommended for dynamic configurations.</p>
Sent by	SwitchManager
Configuration	<i>AllOutOfService</i> (Node = integer, ConnectionID = integer);

Answer Supervision Mode Configure 0x00BB

SwitchKit Name AnswerSuperviseConfig

Type EXS API and SwitchKit API message

Description **Answer Supervision Mode Configure 0x00BB**

Use this message to configure the Answer Supervision mode for a channel or range of channels.

The Answer Supervision Mode options are:

- Propagate Answer to Distant End
- Notify Host of Answer
- Propagate Answer to Distant End and Notify Host of Answer
- No Answer Supervision - no propagation of answer or notification



CAUTION

Do not send this message for channels configured as the following trunk types:

- DP0
- DPT
- FX0 loopstart and groundstart (OK to use if the answer supervision mode is 0x00 or 0x03)

If you use this message for these trunk types, the CSP sends the response *Status* value “Invalid Data Type” (0x17).

Sent by Host

**Example Message
(SwitchKit Socket Log
Output)**

The following socket log output/example message configures the Answer Supervision Mode to propagate answer to distant end for channels 0x00 - 0x17 on spans 0x01 - 0x02.

```
00 12 00 BB 00 00 FF 01 02 0D 03 00 01 00 0D 03 00 02 17 00
```

SwitchKit Code **Configuration**

```
AnswerSuperviseConfig (  
    Node = integer,  
    Range = StartSpan : StartChan - EndSpan : EndChan,  
    AnswerMode = integer);
```

C Structure

```
typedef struct {
    unsigned short StartSpan;
    UBYTE StartChannel;
    unsigned short EndSpan;
    UBYTE EndChannel;
    UBYTE AnswerMode;
} XL_AnswerSuperviseConfig;
```

C++ Class

```
class XLC_AnswerSuperviseConfig : public
    XLC_ChanRangeMessage {
public:
    unsigned short getStartSpan() const;
    void setStartSpan(unsigned short x);
    UBYTE getStartChannel() const;
    void setStartChannel(UBYTE x);
    unsigned short getEndSpan() const;
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00BB)	3, 4	Message Type (0x00BB)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u> Address Method 0x01 - Range	8, 9	Status MSB, LSB
	Number AEs to follow		
	AEs 0x0D Channel (Starting) 0x0D Channel (Ending)	10	Checksum
:	Answer Supervision Mode Enter one of the following values: 0x00 Propagate Answer to Distant End (default) 0x01 Notify Host of Answer 0x02 Propagate Answer to Distant End and Notify Host of Answer 0x03 No Answer Supervision – No propagation of answer or notification FX0 Loopstart/FX0 Groundstart The only valid entries for FX0 loopstart and FX0 groundstart are 0x0000 and 0x0003.		
:	Checksum		

AppConnectionQuery

- Type** SwitchKit API message
- Purpose** Use the *AppConnectionQuery* message to query the Low-Level Communicator (LLC) for description data of a specific application.
- Description** The message is for querying the LLC for description data of a specific application. All of the data is returned in *AppConnectionQueryAck*. In addition to returning the application IDs of connected applications, an acknowledgement will also contain the application IDs of the primary LLC and redundant LLC.
- Sent by** Application
- How to use** To use an *AppConnectionQuery* message construct the message and set the application ID:

```
SKC_AppConnectionQuery acq;
acq.setAppID(app_id);
```
- Arguments** The following table shows the user modifiable argument:

Argument	Description
AppID	Specifies the name of the specific application.

- Status Field** The following table shows the possible return values of the status field:

Value	Description
OK	Query successfully acknowledged
SK_NO_ACK_FROM_SWITCH	CSP did not respond.
SK_UNKNOWN_APP	You used an unknown AppID. This indicates that no application with the specified ID is connected to the LLC.
SK_NO_APP_DATA	Indicates that the application has no description data associated with it.

```

C Structure      typedef struct {
                    int AppID;
                    } SK_AppConnectionQuery;

C Structure Response  typedef struct {
                    int Status;
                    int AppID;
                    char AppName[80];
                    char AppVersion[16];
                    int PID;
                    char Hostname[32];
                    char StartTime[32];
                    char SkVersion[16];
                    char UserData[50];
                    } SK_AppConnectionQueryAck;

C++ Class      class SKC_AppConnectionQuery : public SKC_ToolkitMessage
                    {
                    public:
                    int getAppID() const;
                    void setAppID(int x);
                    };

C++ Class Response  class SKC_AppConnectionQueryAck : public SKC_ToolkitAck
                    {
                    public:
                    int getStatus() const;
                    void setStatus(int x);
                    int getAppID() const;
                    void setAppID(int x);
                    const char *getAppName() const;
                    void setAppName(const char *x);
                    const char *getAppVersion() const;
                    void setAppVersion(const char *x);
                    int getPID() const;
                    void setPID(int x);
                    const char *getHostname() const;
                    void setHostname(const char *x);
                    const char *getStartTime() const;
                    void setStartTime(const char *x);
                    const char *getSkVersion() const;
                    void setSkVersion(const char *x);
                    const char *getUserData() const;
                    void setUserData(const char *x);
                    };

```

AppDescriptionData

Type	SwitchKit API message
Description	<p>The message <i>AppDescriptionData</i> is used by the function <i>sk_appDescriptionData()</i> that provides additional information to help uniquely identify the application.</p> <p>This message does not have a response acknowledgment.</p>
Sent by	Function <code>sk_appDescriptionData()</code>
C Structure	<pre>typedef struct { } SK_AppDescriptionData;</pre>
C++ Class	<pre>class SKC_AppDescriptionData : public SKC_AdminMessage { public: };</pre>

AppPopulationQuery

Type	SwitchKit API message
Purpose	Use the <i>SK_AppPopulationQuery</i> to query the application IDs of all applications connected to the Low-Level Communicator (LLC).
Description	The <i>AppPopulationQuery</i> message requires no arguments. The message is used for querying the application IDs of all applications connected to the LLC. All IDs are returned in the <i>AppPopulationQueryAck</i> message.
Sent by	Application
How to use	To use an <i>AppPopulationQuery</i> , construct the message with no argument: <pre>SKC_AppPopulationQuery apq;</pre>
Status Field	The following table shows the possible return values of the status field:

Value	Description
OK	Query successful acknowledged.
SK_NO_ACK_FROM_SWITCH	The CSP did not respond.

C Structure	<pre>typedef struct { } <i>SK_AppPopulationQuery</i>;</pre>
C Structure Response	<pre>typedef struct { int Status; int NumApps; int AppIDs[50]; } <i>SK_AppPopulationQueryAck</i>;</pre>
C++ Class	<pre>class <i>SKC_AppPopulationQuery</i> : public SKC_ToolkitMessage { public: };</pre>

C++ Class Response

```
class SKC_AppPopulationQueryAck : public SKC_ToolkitAck
{
public:
    int getStatus() const;
    void setStatus(int x);
    int getNumApps() const;
    void setNumApps(int x);
    const int *getAppIDs() const;
    void setAppIDs(const int *x);
};
```

ARP Cache Query 0x00FC

SwitchKit Name ARPCacheQuery

Type EXS API and SwitchKit API message

Description: **ARP Cache Query 0x00FC**

Use this message to query or flush the entries in a module's ARP cache. If a query is sent, the CSP returns the data in an *ARP Cache Report* message. You can query a specific module by using the ARP query TLV. If the amount of data returned exceeds the 512 byte maximum limit of a host message, the CSP sends multiple messages. To remove entries, use the ARP Remove TLV. You can remove multiple entries by including multiple TLVs. The CSP returns a Status of Positive Acknowledgment (0x000A) upon removal.

NOTE: Entries are automatically flushed from the table if not accessed within five minutes after the connection is torn down.

Sent by: Host

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE TLVDataType;
    UBYTE TLVCount;
    UBYTE TLVData[221];
} XL_ARPCacheQuery;
```

C++ Class

```
class XLC_ARPCacheQuery : public XLC_OutboundMessage {
public:
    virtual int getTag() const;
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    // Extended addressing functions
    // Slot AIB functions
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getTLVDataType() const;
    void setTLVDataType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getTLVData() const;
    UBYTE *getTLVData();
    void setTLVData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00FC)	3, 4	Message Type (0x00FC)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs <hr/> Number of AEs to follow <hr/> AE 0x01 Slot	8, 9	Response Status MSB, LSB 0x0001 Invalid VoIP Module Length 0x0002 Invalid Number of TLVs 0x0003 Invalid VoIP Module 0x0004 Invalid Number of Modules 0x0005 Invalid ARP Length 0x0006 Invalid ARP Cache TLV 0x0007 Entry Not Found in ARP Cache 0x0010 Positive ACK (Entry Flushed)
:	Data Type (0x00: TLVs)	10	Checksum
:	Number of TLVs to follow		
:	Data (TLVs) 0x01DC VoIP Module 0x01DD Flush ARP Cache Table Entry		
:	Checksum		

ARP Cache Report 0x00FB

SwitchKit Name ARPCacheReport

Type EXS API and SwitchKit API message

Description: **ARP Cache Report 0x00FB**

This message is sent by the CSP in response to an *ARP Cache Query* message. The report includes the IP address and Ethernet address for each entry queried. If the ARP cache size is larger than 512 bytes, multiple messages are sent.

SwitchKit Users

If you want to get this report, you must register for it by calling the `sk_msgRegister` function with the parameter, `SK_ARP_CACHE_REPORT`.

Sent by: CSP

Resent in EXS API: This message is resent once after five seconds.

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short Status;
    UBYTE AddrInfo[30];
    unsigned short TLVDataType;
    unsigned short TLVCount;
    UBYTE TLVData[217];
} XL_ARPCacheReport;
```

C++ Class

```
class XL_ARPCacheReport : public XLC_InboundMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    // Extended addressing functions
    // Slot AIB functions
    UBYTE getSlot() const;;
    void setSlot(UBYTE x);
    // Module AIB functions
    UBYTE getModule() const;
    void setModule(UBYTE x);
    unsigned short getTLVDataType() const;void
setTLVDataType(unsigned short x);
    unsigned short getTLVCount() const;
    void setTLVCount(unsigned short x);
    const UBYTE *getTLVData() const;
```

```
UBYTE *getTLVData();  
void setTLVData(UBYTE *x);  
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0005)
3, 4	Message Type (0x00FB)	3, 4	Message Type (0x00FB)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Sequence Number
7	Logical Node ID	7	Logical Node ID
8, 9	Query Status (MSB, LSB) Always 0x10 (Positive ACK)	8	Checksum
:	<u>AIB</u> Address Method 0x00 - Individual AEs		
	Number of AEs to follow		
	AE 0x01 Slot		
:	Data Type: TLVs (0x00)		
:	Total ARP Cache Entry Count		
:	Number of TLVs to follow		
:	Data (TLVs) 0x01DC VoIP Module 0x01DE ARP Cache Table Entry 0x01E0 ARP Cache Report Information		
:	Checksum		

Assign EXS Host/Slave 0x006E

SwitchKit Name	AssignEXSHostSlave
Type	EXS API and SwitchKit API message
Description	<p>Assign EXS Host/Slave 0x006E</p> <p>Use this message to establish a relationship between a host node and a slave node in a CSP system. You can route this message through any node.</p> <p>If you indicate the same node as both the host node and the slave node, the CSP returns the response status 0xFC (Invalid Logical Node ID).</p> <p>NOTE: Do not use this message to establish a node as a host node. A node becomes a host node when any API message is sent to it through a direct TCP connection.</p>
Sent by	Host
SwitchKit Code	<p>Configuration</p> <pre>AssignEXSHostSlave(Node = integer, HostNode = integer, SlaveNode = integer);</pre> <p>C Structure</p> <pre>typedef struct { BaseFields Base; UBYTE HostNode; UBYTE SlaveNode; } XL_AssignEXSHostSlave;</pre> <p>C Structure Response</p> <pre>typedef struct { BaseFields Base; unsigned short Status; unsigned short Span; UBYTE Channel; } XL_AssignEXSHostSlaveAck;</pre> <p>C++ Class</p> <pre>class XLC_AssignEXSHostSlave : public XLC_OutboundMessage { public: UBYTE getHostNode() const; void setHostNode(UBYTE x); UBYTE getSlaveNode() const; void setSlaveNode(UBYTE x); };</pre>

C++ Class Response

```
class XLC_AssignEXSHostSlaveAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x006E)	3, 4	Message Type (0x006E)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID (Host Node)	7	Logical Node ID (Host Node)
:	<u>AIB</u> Address Method 0x00 - Individual AEs Number of AEs to follow AE 0x0E EXS Host-Slave Node	8, 9	Response Status MSB, LSB If the value of this field is anything other than a Positive Acknowledgment (0x10), the system inserts the Logical Node AIB after this field.
:	Checksum	:	Only if Response Status field is not 0x10: AE 0x0E EXS Host-Slave Node
		:	Checksum

Assign Logical Node ID 0x0010

SwitchKit Name	AssignNode
Type	EXS API and SwitchKit API message
Description	<p>Assign Logical Node ID 0x0010</p> <p>Use this message to assign a logical node ID to a CSP that is a node in a multi-node system.</p> <p>If the node has an EXNET-ONE card installed or if you plan to have an EXNET-ONE card installed, you must assign the node ID at configuration.</p> <p>The default logical node ID for a stand-alone CSP is 0xFF. To specify the node that you are assigning a logical ID, you use its physical node ID (serial number).</p> <p style="text-align: center;">NOTE: For SwitchKit, DO NOT use the <i>XL_AssignNode</i> message with EXS SwitchKit. <i>XL_AssignNode</i> is obsolete with the introduction of <i>SK_AddLLCNode</i>.</p>
Sent by	Host
Example Message (SwitchKit Socket Log Output)	<p>In the following socket log output/example message, the host assigns logical node ID 01 to the physical node 0000 23AC for the first time:</p> <pre>00 0D 00 10 00 00 FF 00 01 10 05 00 00 23 AC 01</pre>
SwitchKit Code	<p>Configuration</p> <pre>AssignNode (Node = integer, PhysicalNode = integer, LogicalNode = integer);</pre> <p>C Structure</p> <pre>typedef struct { BaseFields Base; int PhysicalNode; UBYTE LogicalNode; } XL_AssignNode;</pre> <p>C++ Class</p> <pre>class XLC_AssignNode : public XLC_OutboundMessage { public: int getPhysicalNode() const; void setPhysicalNode(int x); UBYTE getLogicalNode() const; void setLogicalNode(UBYTE x); UBYTE getISUPRemCon() const; void setISUPRemCon(UBYTE x);</pre>

};

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0010)	3, 4	Message Type (0x0010)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Current Logical Node ID If assigning for first time, byte 7 is 0xFF.	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs Number of AEs to follow AEs 0x10 Logical/Physical Node	8, 9	Status (MSB, LSB) 0x007F Software Module Locked See also the Common Response Status Values chapter
:	ISUP Remote Control 0x00 Disable ISUP Remote Control (Default) 0x01 Enable ISUP Remote Control		
:	Checksum	10	Checksum

Assign Logical Span ID 0x00A8

SwitchKit Name	AssignSpan
Type	EXS API and SwitchKit API message
Description	<p>Assign Logical Span ID 0x00A8</p> <p>Use this message to map a logical span ID to a physical location. You must assign a unique ID to each span in the system before you can configure any spans or channels.</p>
Sent by	Host
Example Message (SwitchKit Socket Log Output)	<p>In the following socket log output/example message, the host assigns spans in a single node system to offset 00 in a VDAC-ONE card.</p> <pre>00 0D 00 A8 00 00 FF 01 00 01 11 04 A8 01 00</pre>
SwitchKit Code	<p>Configuration</p> <pre>AssignSpan (Node = integer, Span = integer, Slot = integer, Offset = integer);</pre> <p>C Structure</p> <pre>typedef struct { UBYTE AddrInfo[30]; UBYTE reserved48[222]; } XL_AssignSpan;</pre> <p>C Structure Response</p> <pre>typedef struct { unsigned short Status; UBYTE reserved6[13]; UBYTE AddrInfo[30]; UBYTE reserved50[220]; } XL_AssignSpanAck;</pre> <p>C++ Class</p> <pre>class XLC_AssignSpan : public XLC_SlotMessage { public: const UBYTE *getAddrInfo() const; UBYTE *getAddrInfo(); void setAddrInfo(UBYTE *x); // Extended addressing functions // Assign Span AIB functions XBYTE getSpan() const;</pre>

```
void setSpan(XBYTE x);
UBYTE getSlot() const;
void setSlot(UBYTE x);
UBYTE getOffset() const;
void setOffset(UBYTE x);
};
```

C++ Class Response

```
class XLC_AssignSpanAck : public XLC_AcknowledgeMessage
{
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    // Extended addressing functions
    // Assign Span AIB functions
    XBYTE getSpan() const;
    void setSpan(XBYTE x)
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getOffset() const;
    void setOffset(UBYTE x);
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00A8)	3, 4	Message Type (0x00A8)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u> Address Method 0x00 - Individual AEs	8, 9	Status MSB, LSB 0x007F - Software Module Locked See also the Common Response Status Values chapter
	Number of AEs to follow	:	<u>AIB</u> Same as message.
	AE 0x11 Logical/Physical Span To de -assign spans, see AIB below table.		Checksum
	Checksum		:

0x11 Logical Span AIB for De-Assigning Spans

To de-assign a physical span, logical span, or all spans, use the field values as shown in the table below in the *Assign Logical Span ID* message (0x00A8).

Byte	AIB Field Description			
0	Address Method: 0x00			
1	Number of AEs: 0x01			
2	Address Element: 0x11			
3	Data Length: 0x04			
		All Spans	Physical Span ID	Logical Span ID
4	Data[0] Logical Span ID, MSB	0xFF	0xFF	Valid Logical Span
5	Data[1] Logical Span ID, LSB	0xFF	0xFF	Valid Logical Span
6	Data[2] Slot	0xFF	Valid Physical Slot	0xFF
7	Data[3] Physical Span	0xFF	Valid Span Offset	0xFF

NOTES:

1. De-assigning a Logical Span ID takes the span, and all channels on the span, out of service.
2. De-assigning a span resets its configuration to the default values. When you want to re-initialize a card's configuration (for example, when you restart a host application) use the *Reset Configuration* message.
3. When you de-assign a span, you do **not** receive a *DSO Status Change* message with a *Channel Status* field value 0x01 (Out of Service) for every channel on the span. Assume that all channels on a de-assigned span are out of service.
4. De-assigning all spans will clean up all virtual slots which acts as if the virtual cards were removed.

Associate Chan Group

Type	SwitchKit API message
Description	<p>Use the <i>SK_AssociateChanGroup</i> message to associate a ChannelGroup with specified channels. <i>SK_AssociateChanGroup</i> commands are cumulative. To build a non-contiguous channel group, issue multiple commands. Furthermore, as long as the Low-Level Communicator (LLC) is running, the channel groups are in force. It is recommended that in your configuration files, you have a <i>SK_ClearChanGroup</i> for all the channel groups you intend to use.</p> <p>Important! In channel management, if an application specifies a channel group that was not configured in LLC using an <i>AssociateChannelGroup</i>, the LLC will create that group and assign no channels to that channel group. When <i>AssociateChannelGroup</i> is sent to the LLC, LLC will add the specified channels to that group.</p> <p>If sent by SwitchManager, and <i>SK_AssociateChanGroup</i> includes any signaling channels (D Channel, Link or R2 signaling channel), SwitchManager will exclude those channels from that group.</p> <p>In case an application is responsible for building the groups of channels, the application must call the function <i>sk_associateChannelGroup()</i>. That function then uses the message to create the specified groups.</p>
Sent by	SwitchManager or Application
Configuration	<pre><i>AssociateChanGroup</i>(ChannelGroup = quoted string, range = startSpan:startChannel - endSpan:endChannel);</pre>
C Structure	<pre>typedef struct { unsigned short StartSpan; unsigned short EndSpan; UBYTE StartChannel; UBYTE EndChannel; char ChannelGroup[45]; } <i>SK_AssociateChanGroup</i>;</pre>
C++ Class	<pre>class <i>SKC_AssociateChanGroup</i> : public SKC_AdminMessage { public: unsigned short getStartSpan() const;</pre>

```
void setStartSpan(unsigned short x);
unsigned short getEndSpan() const;
void setEndSpan(unsigned short x);
UBYTE getStartChannel() const;
void setStartChannel(UBYTE x);
UBYTE getEndChannel() const;
void setEndChannel(UBYTE x);
const char *getChannelGroup() const;
void setChannelGroup(const char *x);
};
```

Example The following is an example of AssociateChanGroup for Group Name “all”, with a range of span 0/channel 0 - span 2/channel 23.

```
AssociateChanGroup(
    groupname = "all",
    range = 0:0-2:23);
```

If span 0/channel 23 were a D Channel, SwitchManager would split the message as follows:

```
AssociateChanGroup(
    groupname = "all",
    range = 0:0-0:22);
```

```
AssociateChanGroup(
    groupname = "all",
    range = 1:0-2:23);
```

B Channel Configure 0x00C8

SwitchKit Name BChannelConfig

Type EXS API and SwitchKit API message

Description **B Channel Configure 0x00C8**

Use this message to configure the following information contained in ISDN PRI messages associated with outgoing B channel calls:

- Network Type (network-specific facilities)
- Outgoing Information Transfer Capability
- Calling Number Type
- Calling Number Plan ID
- Calling Presence Indicator
- Calling Screening Indicated
- Called Number Type
- Called Number Plan ID
- Outgoing Response Enable Values

Related API Messages • *B Channel Query* 0x00CA (SwitchKit Name: BChannelQuery)

Sent by Host

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00C8)	3, 4	Message Type (0x00C8)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB (Starting with Byte 0)	8, 9	Status MSB, LSB
		10	Checksum
:	Entity (B Channel Characteristic)		
:	Configuration Option (Value)		
:	Checksum		

SwitchKit Code Configuration

```
BChannelConfig (
    Node = integer,
    Range = StartSpan:StartChan - EndSpan:EndChan,
    Entity = integer,
    Value = integer);
```

C Structure

```
typedef struct {
    unsigned short StartSpan;
    UBYTE StartChannel;
    unsigned short EndSpan;
    UBYTE EndChannel;
    UBYTE Entity;
    UBYTE Value;
} XL_BChannelConfig;
```

C++ Class

```
class XLC_BChannelConfig : public XLC_ChanRangeMessage {
public:
    unsigned short getStartSpan() const;
    void setStartSpan(unsigned short x);
    UBYTE getStartChannel() const;
    void setStartChannel(UBYTE x);
    unsigned short getEndSpan() const;
    void setEndSpan(unsigned short x);
    UBYTE getEndChannel() const;
    void setEndChannel(UBYTE x);
    UBYTE getEntity() const;
    void setEntity(UBYTE x);
    UBYTE getValue() const;
    void setValue(UBYTE x);
};
```

EXS API Hex Format - Detail

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1	Length, MSB	1	Length, MSB (0x00)
2	Length, LSB (N)	2	Length, LSB (0x07)
3	Message Type, MSB (0x00)	3	Message Type, MSB (0x00)
4	Message Type, LSB (0xC8)	4	Message Type, LSB (0xC8)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number

B Channel Configure 0x00C8

7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u> Address Method 0x01 Range of AEs	8, 9	Status MSB, LSB
	Number of AEs to follow	10	Checksum
	<u>AEs</u> 0x0D Channel (Starting) 0x0D Channel (Ending)		
:	Entity (B Channel Characteristic) Enter the value for the B channel characteristic that you are configuring: 0x01 Network Type (network-specific facilities) 0x02 Outgoing Information Transfer Capability 0x03 Calling Number Type 0x04 Calling Number Plan ID 0x05 Calling Presence Indicator 0x06 Calling Screening Indicated 0x07 Called Number Type 0x08 Called Number Plan ID 0x09 Outgoing Response Enable Values		

:	<p>Configuration Option (Value) Enter the desired configuration option. The options for each B Channel Characteristic are listed below:</p> <p>0x01 Network Type</p> <ul style="list-style-type: none">0x01 Do Not Include Network-Specific Facilities (NSF) IE (default)0x02 AT&T Software Defined Network0x03 AT&T Megacom 800 Service0x04 AT&T Megacom0x05 AT&T Accunet0x06 AT&T Long Distance Service0x07 AT&T International 8000x08 AT&T Multiquest0x09 Northern Telecom Private Net0x0A Northern Telecom Inroads0x0B Northern Telecom OutWats0x0C Northern Telecom Foreign Exchange0x0D Northern Telecom Tie Trunk0x0E Northern Telecom TRO Call0x10 NI 2 INWATS0x11 NI 2 OUTWATS0x12 NI 2 Foreign Exchange0x13 NI 2 Tie Trunk <p>0x02 Outgoing Information Transfer Capability</p> <ul style="list-style-type: none">0x01 Voice (Default)0x02 Modem (3.1 KHz audio)0x03 56 KBPS0x04 64 KBPS0x05 64 KBPS Restricted0x06 Ho (Information Transfer Rate 384 Kbps)0x07 H11 (Information Transfer Rate 1536 Kbps)0x08 Multi-rate <p>0x03 Calling Number Type</p> <ul style="list-style-type: none">0x01 Unknown0x02 International number0x03 National number (Default)0x04 Subscriber number0x05 Abbreviated number <p>0x04 Calling Number Plan ID</p> <ul style="list-style-type: none">0x01 Unknown0x02 ISDN Numbering Plan/Recommendation E.164/E.163 (Default)0x03 Private numbering plan0x04 Telephony numbering plan <p>0x05 Calling Presentation Indicator</p> <ul style="list-style-type: none">0x01 Presentation is allowed (Default)0x02 Presentation is restricted0x03 Number not available due to interworking
---	---

B Channel Configure 0x00C8

	0x06 Calling Screen Indicator
	0x01 User provided, not screened (Default)
	0x02 User provided, verified and passed
	0x03 User provided, verified and failed
	0x04 Network provided
	0x07 Called Number Type
	0x01 Unknown type of number
	0x02 International number
	0x03 National number (Default)
	0x04 Subscriber number
	0x05 Abbreviated number
	0x08 Called Number Plan ID
	0x01 Unknown numbering plan (Default)
	0x02 ISDN Numbering Plan/Recommendation E.164/E.163
	0x03 Private numbering plan
	0x04 Telephony numbering plan
	0x09 Outgoing Response Enable (This field is reserved for future use)
:	Checksum

B Channel Query 0x00CA

SwitchKit Name BChannelQuery

Type EXS API and SwitchKit API message

Description **B Channel Query 0x00CA**

Use this message to query the configuration and current channel state information on an ISDN PRI B Channel, DASS2, or DPNSS B Channel. The information in the response varies for ISDN and DASS2/DPNSS.

Related API Messages *B Channel Configure 0x00C8* (SwitchKit Name: BChannelConfig)

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {  
    unsigned short Span;  
    UBYTE Channel;  
} XL_BChannelQuery;
```

C Structure Response

```
typedef struct {  
    unsigned short Status;  
    unsigned short Span;  
    UBYTE Channel;  
    UBYTE DChannelID;  
    UBYTE Facility;  
    UBYTE BChannel;  
    UBYTE CurrentState;  
    UBYTE PriDEnable;  
    UBYTE PriBEnable;  
    UBYTE ExternalEnable;  
    UBYTE HostEnable;  
    UBYTE ControllingDChannelFlag;  
    UBYTE NetworkType;  
    UBYTE CallType;  
    UBYTE CallingType;  
    UBYTE CallingNumberPlanID;  
    UBYTE CallingPresentationInd;  
    UBYTE CallingScreeningInd;  
    UBYTE CalledType;  
    UBYTE CalledNumberPlanID;  
    UBYTE PCMEncodingFormat;  
    UBYTE DistantEndReleaseMode;  
    UBYTE AnswerSupervisionMode;  
    UBYTE ISDNEventEnable;  
    UBYTE LocalEndReleaseMode;  
    UBYTE NetworkBlocked;  
} XL_BChannelQueryAck;
```

C++ Class

```
class XLC_BChannelQuery : public XLC_OneChannelOutbound {
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
};
```

C++ Class Response

```
class XLC_BChannelQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getDChannelID() const;
    void setDChannelID(UBYTE x);
    UBYTE getFacility() const;
    void setFacility(UBYTE x);
    UBYTE getBChannel() const;
    void setBChannel(UBYTE x);
    UBYTE getCurrentState() const;
    void setCurrentState(UBYTE x);
    UBYTE getPriDEnable() const;
    void setPriDEnable(UBYTE x);
    UBYTE getPriBEnable() const;
    void setPriBEnable(UBYTE x);
    UBYTE getExternalEnable() const;
    void setExternalEnable(UBYTE x);
    UBYTE getHostEnable() const;
    void setHostEnable(UBYTE x);
    UBYTE getControllingDChannelFlag() const;
    void setControllingDChannelFlag(UBYTE x);
    UBYTE getNetworkType() const;
    void setNetworkType(UBYTE x);
    UBYTE getCallType() const;
    void setCallType(UBYTE x);
    UBYTE getCallingType() const;
    void setCallingType(UBYTE x);
    UBYTE getCallingNumberPlanID() const;
    void setCallingNumberPlanID(UBYTE x);
    UBYTE getCallingPresentationInd() const;
    void setCallingPresentationInd(UBYTE x);
    UBYTE getCallingScreeningInd() const;
    void setCallingScreeningInd(UBYTE x);
    UBYTE getCalledType() const;
    void setCalledType(UBYTE x);
    UBYTE getCalledNumberPlanID() const;
    void setCalledNumberPlanID(UBYTE x);
    UBYTE getPCMEncodingFormat() const;
    void setPCMEncodingFormat(UBYTE x);
    UBYTE getDistantEndReleaseMode() const;
    void setDistantEndReleaseMode(UBYTE x);
```

B Channel Query 0x00CA

```
UBYTE getAnswerSupervisionMode() const;  
void setAnswerSupervisionMode(UBYTE x);  
UBYTE getISDNEventEnable() const;  
void setISDNEventEnable(UBYTE x);  
UBYTE getLocalEndReleaseMode() const;  
void setLocalEndReleaseMode(UBYTE x);  
UBYTE getNetworkBlocked() const;  
void setNetworkBlocked(UBYTE x);  
};
```

Overview of message The following two tables provides an overview of this message. The tables following them, provides the detail for each byte.

Overview of ISDN PRI B Channels

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00CA)	3, 4	Message Type (0x00CA)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs	8, 9	Status MSB, LSB
	Number of AEs to follow		
	0x0D Channel		
:	Checksum	:	AIB Address Method 0x00 - Individual AEs
			Number of AEs to follow
			0x0D Channel
		:	D Channel ID (Reserved)
(Response continued below.)			
:	Facility Number		
:	B Channel ID (Reserved)		
:	Current State (Reserved)		
:	PRI D Enable (Reserved)		
:	PRI B Enable (Reserved)		
:	External Enable (Reserved)		
:	Host Enable (Reserved)		
:	Controlling D Channel Flag		
:	Network Type		
:	Outgoing Information Transfer Capability		
:	Calling Number Type		
:	Calling Number Plan ID		
:	Calling Presentation Indicator		
:	Calling Screen Indicator		
:	Called Number Type		
:	Called Number Plan ID		
:	PCM Encoding Format		
:	Distant End Release Mode		
:	Answer Supervision Mode		

B Channel Query 0x00CA

:	ISDN Event Enable (Reserved)
:	Local End Release Mode
:	Network Blocked (Reserved)
:	Checksum

**Overview of DASS2/DPNSS
B Channels**

MESSAGE		RESPONSE	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1	Length, MSB	1	Length, MSB
2	Length, LSB (N)	2	Length, LSB (N)
3	Message Type, MSB (0x00)	3	Message Type, MSB (0x00)
4	Message Type, LSB (0xCA)	4	Message Type, LSB (0xCA)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u> Address Method 0x00 - Individual AEs Number of AEs to follow 0x0D Channel	8,9	Status MSB, LSB
:	Checksum	:	<u>AIB</u> Address Method 0x00 - Individual AEs Number of AEs to follow 0x0D Channel
(Response continued below.)			
:	D Channel ID (Reserved)		
:	Reserved		
:	B Channel ID (Reserved)		
:	L3P State		
:	L3 State		
:	L2 State		
:	Reserved		
:	Reserved		
:	Controlling D Channel Flag		
:	Reserved		
:	Reserved		
:	Reserved		
:	Reserved		
:	Reserved		
:	Reserved		
:	Reserved		
:	Reserved		
:	Reserved		

B Channel Query 0x00CA

:	PCM Encoding Format
:	Distant End Release Mode
:	Answer Supervision Mode
:	Reserved
:	Local End Release Mode
:	Reserved
:	Checksum

**EXS API Hex Format for
ISDN PRI B Channels -
Detail**

NOTE: The reserved fields are used by Dialogic Technical Support for diagnostic purposes only. Please ignore these fields during normal operation.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00CA)	3, 4	Message Type (0x00CA)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB (Individual AEs) 0x0D Channel	8, 9	Status MSB, LSB
:	Checksum	10	AIB (Individual AEs) 0x0D Channel
		:	D Channel ID (Reserved)
(Response continued below.)			
:	Facility Number The network-relative facility (or span) number for the associated D channel. 0x00: For both FAS and NFAS, this value represents the facility in which the controlling D channel resides 0x01–0x09: For NFAS, these values represent the other facilities that the D channel controls In Facility-Associated Signaling (FAS), one D channel controls the B channels within the same span. In Non-Facility Associated Signaling (NFAS), one D channel controls the B channels on up to 10 spans. These spans are designated as facilities 0–9.		
:	B Channel ID (Reserved)		
:	Current State (Reserved)		
:	PRI D Enable (Reserved) The system derives the value from the state as indicated in the Current State field: 0x00 Not Enabled 0x01 Enabled		
:	PRI B Enable (Reserved) The system derives the value from the state as indicated in the Current State field: 0x00 Not Enabled 0x01 Enabled		
:	External Enable (Reserved) The system derives the value from the state as indicated in the Current State field: 0x00 Not Enabled 0x01 Enabled		
:	Host Enable (Reserved) The system derives the value from the state as indicated in the Current State field: 0x00 Not Enabled 0x01 Enabled		

B Channel Query 0x00CA

:	Controlling D Channel Flag 0x00 Channel is not a D channel 0x01 Channel is a D channel
:	Network Type 0x01 Do Not Include Network-Specific Facilities (NSF) IE 0x02 AT&T Software Defined Network 0x03 AT&T Megacom 800 Service 0x04 AT&T Megacom 0x05 AT&T Accunet 0x06 AT&T Long Distance Service 0x07 AT&T International 800 0x08 AT&T Multiquest 0x09 Northern Telecom Private Net 0x0A Northern Telecom Inroads 0x0B Northern Telecom OutWats 0x0C Northern Telecom Foreign Exchange 0x0D Northern Telecom Tie Trunk 0x0E Northern Telecom TRO Call
:	Outgoing Information Transfer Capability 0x01 Voice 0x02 Modem 0x03 56 Kbps 0x04 64 Kbps 0x05 64 Kbps Restricted
:	Calling Number Type 0x01 Unknown 0x02 International number 0x03 National number 0x04 Subscriber number 0x05 Abbreviated number
:	Calling Number Plan ID 0x01 Unknown 0x02 ISDN Numbering Plan/Recommendation E.164/E.163 0x03 Private numbering plan 0x04 Telephony numbering plan
:	Calling Presentation Indicator 0x01 Presentation is allowed 0x02 Presentation is restricted 0x03 Number not available due to interworking
:	Calling Screen Indicator 0x01 User provided, not screened 0x02 User provided, verified and passed 0x03 User provided, verified and failed 0x04 Network provided
:	Called Number Type 0x01 Unknown type of number 0x02 International number 0x03 National number 0x04 Subscriber number 0x05 Abbreviated number

B Channel Query 0x00CA

:	Called Number Plan ID 0x01 Unknown numbering plan 0x02 ISDN numbering plan/Recommendation E.164/E.163 0x03 Private numbering plan 0x04 Telephony numbering plan
:	PCM Encoding Format 0x01 μ -law Encoding 0x02 A-law Encoding
:	Distant End Release Mode 0x01 Park 0x02 Release
:	Answer Supervision Mode 0x00 Propagate Answer to the Distant End 0x01 Notify Host of Answer 0x02 Propagate Answer to the Distant End and Notify Host of Answer 0x03 No Answer Supervision – no propagation of answer or notification
:	ISDN Event Enable (Reserved) You can send this data to the host using the PPL messages.
:	Local End Release Mode 0x01 Park 0x02 Release
:	Network Blocked (Reserved)
:	Checksum

**EXS API Hex Format for
DASS2/DPNSS B Channels
- Detail**

NOTE: The reserved fields are used by Dialogic Technical Support for diagnostic purposes only. Please ignore these fields during normal operation.

MESSAGE		RESPONSE	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1	Length, MSB	1	Length, MSB
2	Length, LSB (N)	2	Length, LSB (N)
3	Message Type, MSB (0x00)	3	Message Type, MSB (0x00)
4	Message Type, LSB (0xCA)	4	Message Type, LSB (0xCA)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB (Individual AEs) 0x0D Channel	8, 9	Status MSB, LSB
:	Checksum	10	AIB (Individual AEs) 0x0D Channel
(Response continued below.)			
:	D Channel ID (Reserved)		
:	Reserved		
:	B Channel ID (Reserved)		
:	L3P State - Call Control PPL State (0x0D) Indicates the state of the DASS2/DPNSS Call Control PPL component.		

B Channel Query 0x00CA

:	<p>L3 State</p> <p>DASS2 States</p> <ul style="list-style-type: none"> 0x01 Connected 0x02 Dial In (receiving overlap call) 0x03 Dial Out (sending overlap call) 0x04 Idle 0x05 Incoming (received CS) 0x06 Offered (received ICI[C]) 0x07 Offered Incomplete (received ICI[I]) 0x08 Origination (sent ISRM) 0x09 Recovery (Received unexpected message in Idle State) 0x0A Releasing (Waiting for CIM) 0x0B Routing In 0x0C Routing Out <p>DPNSS States</p> <ul style="list-style-type: none"> 0x01 Channel Idle 0x02 Spurious Message Received 0x03 Incoming Service Request Incomplete 0x04 Incoming Service Request Complete 0x05 Channel Release Guard 0x06 Channel Release Guard Tell Host 0x07 Setup Complete 0x08 Outgoing Service Request Complete 0x09 Outgoing Service Request Incomplete 0x0A Host Release Wait
:	<p>L2 State</p> <ul style="list-style-type: none"> 0x01 Establishing Wait (sent SABMR, waiting for UA) 0x02 Idle (Down) 0x03 Ready (Up) 0x04 Wait Response (Up, waiting for UI[R])
:	Reserved
:	Reserved
:	<p>Controlling D Channel Flag</p> <ul style="list-style-type: none"> 0x00 Channel is not a D channel 0x01 Channel is a D channel
:	Reserved
:	Reserved
:	Reserved
:	Reserved
:	Reserved
:	Reserved
:	Reserved
:	<p>PCM Encoding Format</p> <ul style="list-style-type: none"> 0x01 μ-law Encoding 0x02 A-law Encoding
:	<p>Distant End Release Mode</p> <ul style="list-style-type: none"> 0x01 Park 0x02 Release

B Channel Query 0x00CA

:	Answer Supervision Mode 0x00 Propagate Answer to the Distant End 0x01 Notify Host of Answer 0x02 Propagate Answer to the Distant End and Notify Host of Answer 0x03 No Answer Supervision – no propagation of answer or notification
:	Reserved
:	Local End Release Mode 0x01 Park 0x02 Release
:	Reserved
:	Checksum

Become Active 0x00A1

SwitchKit Name ActivateMatrix

Type EXS API and SwitchKit API message

Description **Become Active 0x00A1**

For EXS API

Use this message to initiate a switchover in an CSP with a redundant CSP Matrix Series 3 Card or SS7 card. You must send this message to the standby CSP Matrix Series 3 Card from the matrix host or send this message to the standby SS7 card from the SS7 local host. If you send this message to the active CSP Matrix Series 3 Card or active SS7 card, it has no effect on the CSP.

Dialogic recommends sending this message from the Matrix host to initiate a switchover to the standby CSP Matrix Series 3 Card or sending this message from the SS7 local host to initiate a switchover to the standby SS7 card.

For SwitchKit API

ActivateMatrix() initiates a Matrix switchover regardless of what parameters are passed in. With EXS SwitchKit you must send a function call to **sk_activateExnetMatrix()**. You cannot send the message directly to the standby CSP Matrix Series 3 Card.

EXS SwitchKit API does not support sending this message from the host to the standby SS7 card.

Sent by Host

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x0005)	1, 2	Length (0x0007)
3, 4	Message Type (0x00A1)	3, 4	Message Type (0x00A1)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Checksum	8, 9	Status MSB, LSB
		10	Checksum

BroadcastLoad

Type	EXS SwitchKit API message
Purpose	Do not use this message. This is an internal message used by the function call <i>sk_broadcastLoad()</i> , refer to the <i>EXS SwitchKit Programmer's Guide</i> for information on the <i>sk_broadcastLoad()</i> function.
Description	The Low-Level Communicator (LLC) uses the load value to determine how to distribute the load across several applications that are load sharing.
Sent by	Function <i>sk_broadcastLoad()</i>

Busy Out 0x0018

SwitchKit Name BusyOut

Type EXS API and SwitchKit API message

Description **Busy Out 0x0018**

Use this message to “busy out” a channel or range of channels. This allows the host to control the incoming call rate by busying out a selected group of channels, normally on the front end of a system.

You can busy out channels configured with the following trunk types only:

- E&M
- FXO Loop Start
- FXO Ground Start

Related API Messages *Busy Out Flag Configure* 0x00D3 (SwitchKit Name: BusyOutFlagConfig)

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short StartSpan;
    UBYTE StartChannel;
    unsigned short EndSpan;
    UBYTE EndChannel;
    UBYTE Action;
} XL_BusyOut;
```

C++ Class

```
class XL_C_BusyOut : public XLC_ChanRangeMessage {
public:
    unsigned short getStartSpan() const;
    void setStartSpan(unsigned short x);
    UBYTE getStartChannel() const;
    void setStartChannel(UBYTE x);
    unsigned short getEndSpan() const;
    void setEndSpan(unsigned short x);
    UBYTE getEndChannel() const;
    void setEndChannel(UBYTE x);
    UBYTE getAction() const;
    void setAction(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0018)	3, 4	Message Type (0x0018)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u> Address Method 0x01 - Range Number of AEs to follow 0x0D Channel (Starting) 0x0D Channel (Ending)	8, 9	Status (MSB, LSB)
:	Action 0x01 Busy Out 0x02 Release Busy Out	10	Checksum
:	Checksum		

Busy Out Flag Configure 0x00D3

SwitchKit Name BusyOutFlagConfig

Type EXS API and SwitchKit API message

Description **Busy Out Flag Configure 0x00D3**

Use this message to enable the busy out feature on a channel or range of channels.

You cannot “busy out” a channel with the *Busy Out* message if the *Flag* field in this message is disabled (0x00). Only channels configured with a trunk type of E&M, FXO Loop Start, or FXO Ground Start can be busied out.

Related API Messages *Busy Out* 0x0018 (SwitchKit Name: BusyOut)

Sent by Host

SwitchKit Code **Configuration**

```
BusyOutFlagConfig (
    Node = integer,
    Range = StartSpan:StartChan - EndSpan:EndChan,
    Flag = integer);
```

C Structure

```
typedef struct {
    unsigned short StartSpan;
    UBYTE StartChannel;
    unsigned short EndSpan;
    UBYTE EndChannel;
    UBYTE Flag;
} XL_BusyOutFlagConfig;
```

C++ Class

```
class XLC_BusyOutFlagConfig : public XLC_ChanRangeMessage
{
public:
    unsigned short getStartSpan() const;
    void setStartSpan(unsigned short x);
    UBYTE getStartChannel() const;
    void setStartChannel(UBYTE x);
    unsigned short getEndSpan() const;
    void setEndSpan(unsigned short x);
    UBYTE getEndChannel() const;
    void setEndChannel(UBYTE x);
    UBYTE getFlag() const;
    void setFlag(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0005)
3, 4	Message Type (0x00D3)	3, 4	Message Type (0x00D3)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB	8, 9	Status MSB, LSB
	Address Method 0x00 - Individual AEs	10	
	Number of AEs to follow		
	AEs 0x0D Channel (Starting) 0x0D Channel (Ending)		
:	Flag 0x00 Busy Out Disabled (Default) 0x01 Busy Out Enabled		
:	Checksum		

Call Control Instructions Query 0x0087

SwitchKit Name	CallControlInstructionQuery
Type	EXS API and SwitchKit API message
Description	<p>Call Control Instructions Query 0x0087</p> <p>Use this message to query the preprogrammed inseize and outseize call control instructions on a channel or range of channels. The response lists the instructions using three bytes each for 20 inseize and 20 outseize instructions, even if they are not all configured.</p>
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short Span;
    UBYTE Channel;
} XL_CallControlInstructionsQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE Data[120];
} XL_CallControlInstructionsQueryAck;
```

C++ Class

```
class XLC_CallControlInstructionsQuery : public
    XLC_OneChannelOutbound {
public:
    unsigned short getSpan() const
    void setSpan(unsigned short x)
    UBYTE getChannel() const
    void setChannel(UBYTE x);
};
```

C++ Class Response

```
class XLC_CallControlInstructionsQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x0087)	3, 4	Message Type (0x0087)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs	8, 9	Status MSB, LSB
	Number of AEs to follow		
	AEs 0x0D Channel		
:	Checksum	10	Inseize Instruction 1: Type
		11	Inseize Instruction 1: Data[0]
		12	Inseize Instruction 1: Data[1]
		:	:
		:	Inseize Instruction 20: Type
		:	Inseize Instruction 20: Data[0]
		:	Inseize Instruction 20: Data[1]
		:	Outseize Instruction 1: Type
		:	Outseize Instruction 1: Data[0]
		:	Outseize Instruction 1: Data[1]
		:	:
		:	Outseize Instruction 20: Type
		:	Outseize Instruction 20: Data[0]
		:	Outseize Instruction 20: Data[1]
		:	Checksum

Call Processing Event 0x002E

SwitchKit Name	CallProcessingEvent
Type	EXS API and SwitchKit API message
Description	<p>Call Processing Event 0x002E</p> <p>The CSP uses this message to report a call processing event on a specified channel. The event to be reported is specified by the host in the messages that support the generation of call processing events.</p>
Sent by	CSP
Resent in EXS API	This message is resent once after five seconds.

SwitchKit Code C Structure

```
typedef struct {
    UBYTE AddrInfo[30];
    unsigned short Span;
    UBYTE Channel;
    UBYTE Event;
    UBYTE Data[222];
} XL_CallProcessingEvent;
```

C++ Class

```
class XLC_CallProcessingEvent : public
    XLC_OneChannelMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getEvent() const;
    void setEvent(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE		RESPONSE	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0005)
3, 4	Message Type (0x002E)	3, 4	Message Type (0x002E)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB (starting with byte 0)	8	Checksum
:	Call Processing Event		
:	Data (0)		
:	:		
:	Data (n)		
:	Checksum		

EXS API Hex Format - Detailed

MESSAGE		RESPONSE	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0005)
3, 4	Message Type (0x002E)	3, 4	Message Type (0x002E)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs Number of AEs to follow AEs One of the following: 0x0D Channel 0x55 Conference ID 0x45 Child Conference ID	8	Checksum

Call Processing Event 0x002E

:	Call Processing Event
	0x00* No Event (the specified event did not occur before the Report Call Processing Event instruction was invoked)
	0x01* ANI Request Off-hook
	0x02 Digits
	0x03* Wink 1
	0x04* Wink 2
	0x05* Wink 3
	0x06* Wink 4
	0x07* Wink 5
	0x08* Wink 6
	0x09* Wink 7
	0x0A* Wink 8
	0x0B* Dialtone Detected
	0x0C Reserved
	0x0D Backward R2 Signal
	0x0E First Digit Report
	0x0F PPL Event
	0x10 Digit With Timing Information
	0x11* Receive Idle
	0x13* FAX Tone Detection
	0x20* Answer
	0x21* Tone Complete
	0x22* Outpulsing Complete
	0x23* Flash Detected
	0x24* CPC Detected
	0x25 Recorded Announcement Started
	0x26 Recorded Announcement Complete
	0x27* Timed-out waiting for energy
	0x28 Energy Result Report
	0x29* Coin Tone Detection Timeout
	0x2A Coin Tone Detection Report
	0x2B Play File Started
	0x2C Play File Complete
	0x2D Record File Started
	0x2E Record File Complete
	0x2F File Not Found
	0x30 File Play Underrun
	0x31 File Record Overrun
	0x32 Record Initial Silence Timeout
	0x33 Play File Queue Failure
	0x34 Play File Stopped
	0x35 Record File Stopped
	0x36 DSP Resource Inconsistency
	0x40* DTMF Subscription Terminated (either by remote end or because of subscription refresh failure.)
	0x41 DSP Series 2 Fax Processing Events
	0x42 Truly Silent Channel (PVD/AMD)
	0x43 Voice Detect Rising Edge (PVD/AMD)
	0x44 Voice Detect Falling Edge (PVD/AMD)
	0x45 Long Term Voice Detect Rising Edge (PVD/AMD)
	0x46 Long Term Voice Detect Falling Edge (PVD/AMD)
	0x47 Tone Detect Rising Edge (PVD/AMD)

Call Processing Event 0x002E

	<p>0x48 Tone Detect Falling Edge (PVD/AMD)</p> <p>0x49 Echo Cancellor Event (when a G.164 tone disabling event occurs on the near or far end)</p> <p>0x50 DSP Series 2 FSK Events</p> <p>0xffff</p> <p>*Event includes no data for Data fields</p>
:	<p>Data</p> <p>These fields provide the specific information for each Event. The length and contents of the information depend on the Event. Some events contain no data.</p> <p>Valid entries for this field are listed below by <i>Call Processing Event</i> field value.</p> <p>0x02 Digits</p> <p> Data[0] Number of Stages Reported</p> <p> Data[1] Impulsed Stage Number</p> <p> Data[2] Impulsed Stage Status</p> <p> Data[3] Stage String Count</p> <p> Data[4] String 1: Digit Count</p> <p> Data[5] String 1: First BCD Encoded Digit Pair</p> <p> :</p> <p> Data[n] String 1: Last BCD Encoded Digit Pair</p> <p> Data[n+1] String 2: First BCD Encoded Digit Pair</p> <p> :</p> <p> Data[m] String 2: Last BCD Encoded Digit Pair</p> <p> Data[4–n] are repeated for each string reported.</p> <p> Data[1–n] are repeated for each stage reported.</p> <p>Stage Status</p> <p>The system reports the stage status for each of the four stages. Timers can be configured.</p> <p>Valid entries for this field are as follows:</p> <p>0x10 Positive Acknowledgment - Digits received</p> <p>0x80 Partial Dial Condition - Receive Inter-digit Duration timer expired</p> <p>0x81 Permanent Signal Condition - Receive 1st Digit Detection timer expired</p> <p>0x91 Impulse Stage Not Collected - Stage not collected; no further stage information will be reported.</p> <p>0x92 Digit Complete Timeout</p> <p>0x97 Positive Acknowledgment - Pulse Digits received</p> <p>If the <i>Call Processing Event</i> message is in response to a <i>DSP Service Request</i> or <i>Collect Digit String</i> message, the Impulsed Stage Number is 0xFF. When the digits are collected using a KP_ST framed digit string collection method, the KP_ST digits are not contained in the reported string.</p> <p>0x0D Backward R2 Signal</p> <p> Data[0] R2 Signal Value (0x01–0x0F)</p> <p>0x0E First Digit Report</p> <p> Data[0] DTMF/MFR1 Digit Value (0x00–0x0F)</p>

Call Processing Event 0x002E

0x0F	PPL Event
Data[0]	PPL Event Number (0x00–0xFF)
0x10	Digit with Timing Information. An asterisk (*) denotes a byte that is always set to “1”
Data[0]	Number of Stages Reported *
Data[1]	Impulsed Stage Number
Data[2]	Impulsed Stage Status
Data[3]	Stage String Count *
Data[4]	Digit Count *
Data[5]	BCD-encoded Digit
Data[6, 7]	Digit Duration (in 10 ms units)

The Impulsed Stage Number will be 0xFF if the *Call Processing Event* message is in response to a *DSP Service Request* or *Collect Digit String* message. The maximum time detected for digit duration is 26.35 seconds. When actual digit duration is greater than 26.35 seconds, it reports as 26.35 (0x0A4B) seconds. Only a *DSP Service Request* message request can result in the Call Processing Event of "Digit With Timing Information" being returned by the CSP.

- 0x25 Recorded Announcement Started
 - Data[0,1] ID of announcement being played (MSB, LSB)

- 0x26 Recorded Announcement Completed
 - Data[0,1] ID of announcement completed (MSB, LSB)

- 0x28 Energy Result Report
 - Data[0] Energy Result Flag
 - 0x00 No Energy Detected
 - 0x01 Energy Detected
 - Data[1,2] Duration of previous Energy Result in 10 ms units

- 0x2A Coin Tone Detection Report
 - Data[0,1] Value of coin detected, in cents
 - 0x0005 5 cents (Nickel)
 - 0x000A 10 cents (Dime)
 - 0x0019 25 cents (Quarter)
 - 0x0064 100 cents (Dollar)

- 0x2B Play File Started
 - Data[0-3] File ID (MSB, ... , ... , LSB)
 - Data[4-7] Offset (starting byte)
 - Data[8-9] Server ID

- 0x2C Play File Complete
 - Data[0-3] File ID (MSB, ... , ... , LSB)
 - Data[4-7] Offset (ending byte)
 - Data[8-9] Server ID

- 0x2D Record File Started
 - Data[0-3] File ID (MSB, ... , ... , LSB)
 - Data[4-7] Offset (starting byte)
 - Data[8-9] Server ID

- 0x2E Record File Complete
 - Data[0-3] File ID (MSB, ... , ... , LSB)
 - Data[4-7] Offset Duration of Record (seconds)
 - Data[8-9] Server ID

- 0x2F File Not Found
 - Data[0-3] File ID (MSB, ... , ... , LSB)

- 0x30 File Play Underrun
 - Data[0-3] File ID (MSB, ... , ... , LSB)

- 0x31 File Record Overrun
 - Data[0-3] File ID (MSB, ... , ... , LSB)

0x32	Record Initial Silence Timeout Data[0-3] File ID (MSB, ... , ... , LSB)
0x33	Play File Queue Failure Data[0-3] File ID (MSB, ... , ... , LSB) Data[4-7] Offset (starting byte) Data[8-9] Server ID
0x34	Play File Stopped Data[0-3] File ID (MSB, ... , ... , LSB) Data[4-5] Stop Reason 0x0000 - Channel Released 0x0001 - Host Initiated Stop 0x0004 - Out of Chip Cache Space 0x0005 - Digit Detected Data[6-9] Duration of Play (seconds) Data[10-11] Server ID
0x35	Record File Stopped Data[0-3] File ID (MSB, ... , ... , LSB) Data[4-5] Stop Reason 0x0000 - Channel Released 0x0001 - Host Initiated Stop 0x0002 - Out of File Space 0x0003 - NFS Write Error 0x0004 - Out of Chip Cache Space 0x0005 - Digit Detected Data[6-9] Duration of Record (seconds) Data[10-11] Server ID - For cached files, the Server ID is not applicable. In those cases, the Server ID is 0xFFFF.
0x36	DSP Resource Inconsistency Data[0-3] File ID (MSB, ... , ... , LSB)
0x41	DSP Series 2 Fax Processing Events Data[0, 1] Fax processing event code (see below) Data[2, n] Fax processing event data For Data[0, 1] Fax processing event code, above: <u>Phase A, B, C, D, or E:</u> 0x09 Got CRC error in V.21 frame 0x0F Transmit V21 complete 0x1D ERR frame received 0x35 HS modem detected V21. Switch to receive V21 0x39 Unknown V21 frame received with correct CRC 0x3C CRP frame received 0x12D STAT_PHASE_A 0x12E STAT_PHASE_B 0x12F STAT_PHASE_C 0x130 STAT_PHASE_D 0x131 STAT_PHASE_E

Phase A - Call Establishment

Set Event Bit 0x01

0x3E Request the operator line

Phase B - Pre-Message Procedure

Note: Phase B consists of two sections: Identification and Command

Set Event Bit 0x02

0x06 DCS frame received
0x07 DIS frame received
0x08 DTC frame received
0x0C Switch to ECM mode
0x11 CFR frame received
0x12 CTC frame received
0x1E FTT frame received
0x2F Remote machine can receive detected
0x30 Remote machine can transmit detected
0x36 Receive training ended successfully
0x37 Receive training ended in failure
0x38 Transmit training ended
0x3D CTR frame received
0x42 Switch from ECM mode to Normal mode
0x43 Change mode after EOM

Phase C

Note: Phase C consists of two sections: Phase C1, In-Message Procedure and Phase C2, Message Transmission

Set Event Bit 0x04

0x00 Command to start receiving
0x01 Command to start transmitting
0x15 EOR frame received
0x16 EOR_EOM frame received
0x17 EOR_EOP frame received
0x18 EOR_MPS frame received
0x19 EOR_NULL frame received
0x1A EOR_PRI_EOM frame received
0x1B EOR_PRI_EOP frame received
0x1C EOR_PRI_MPS frame received
0x3A Start of image receive
0x40 ECM frames where detected in error
0x47 End of image transmit
0x48 Start of image transmit
0x4A Receive image operation complete
0x4C Go to point D in the T.30 flow chart
0x4D Got the 4th PPR
0x4E Decided to continue to correct after 4th PPR

Phase D - Post Message Procedure

Set Event Bit 0x08

0x13	EOM frame received
0x14	EOP frame received
0x20	MCF frame received
0x21	MPS frame received
0x22	PIN frame received
0x23	PIP frame received
0x24	PPR frame received
0x25	PPS_EOM frame received
0x26	PPS_EOP frame received
0x27	PPS_MPS frame received
0x28	PPS_NULL frame received
0x29	PPR_PRI_EOM frame received
0x2A	PPS_PRI_EOP frame received
0x2B	PPS_PRI_MPS frame received
0x2C	PRI_EOM frame received
0x2D	PRI_EOP frame received
0x2E	PRI_MPS frame received
0x31	RNR frame received
0x32	RR frame received
0x33	RTN frame received
0x34	RTP frame received
0x41	Receiver is not ready to continue
0x44	Got end of a page
0x45	Last document in the queue was processed

Phase E - Call Release

Set Event Bit 0x10

0x05	DCN frame received
0x0A	Go to B point in T.30 flow chart
0x1F	Transition to C-point on T.30 flow charts
0x65	FAX Protocol Complete only for Send FAX
0xFFF0	FAX I/O Complete for RCV FAX and Send FAX
0xFE	FAX Failed

Timer Event

Set Event Bit 0x20

0x03	Hardware failure timer expired
0x0D	T1 timer expired
0x0E	T4 timer expired
0x3B	T2 timer expired (see T.30)
0x4B	T3 timer expired
0x4F	T5 timer expired

Data Pump

Set Event Bit 0x40

0x02	T.30 is ready to receive
0x04	T.30 is ready to transmit
0x10	Hardware close completed
0x51	V.21 receiver has started to send preamble
0x52	V.21 receiver has completed operation
0x53	V.21 transmitter has started to send preamble
0x54	V.21 transmitter has completed operation
0x55	High-speed receiver has started to send preamble
0x56	High-speed receiver has received TCF, no errors
0x57	High-speed receiver has received TCF with errors
0x58	High-speed transmitter has started to send TCF
0x59	High-speed transmitter has completed TCF
0x5A	High-speed transmitter has started
0x5B	High-speed transmitter has completed operation
0x5C	High-speed receiver has started to send preamble
0x5D	High-speed receiver has lost carrier
0x5E	V21 Receive Frame
0x5F	V21 Transmit Frame
0x62	V21 signal detected by the HS receiver
0x63	Modem subsystem ready to transmit
0x64	Modem subsystem ready to receive
0x65	Modem subsystem has stopped operations
0x79	Non-Standard facilities (NSF) frame sent or receive
0x7A	Non-Standard facilities Set-up (NSS) frame sent or receive
0x7B	Non-Standard facilities Command (NSC) frame sent or receive
0x7C	Called Subscriber Identification (CSI) frame sent or receive
0x7D	Transmitting Subscriber Identification (TSI) frame sent or receive
0x7E	Calling Subscriber Identification (CIG) frame sent or receive
0x7F	Selective Polling (SEP) frame sent or receive
0x80	Subaddress (SUB) frame sent or receive
0x81	Password (PWD) frame sent or receive
0x82	Digital Identification Signal (DIS) frame sent or received
0x83	Digital Command Signal (DCS) frame sent or receive

Call Processing Event 0x002E

0x84	Digital Transmit Command (DTC) frame sent or receive
0x85	Continue To Correct (CTC) frame sent or receive
0x86	Partial Page Request (PPR) frame sent or received
0x87	Partial Page Signal-NULL(PPS-NULL) frame sent or receive
0x88	Partial Page Signal (PPS)-MPS frame sent or receive
0x89	Partial Page Signal (PPS)-EOM frame sent or receive
0x8A	Partial Page Signal (PPS)-EOP frame sent or receive
0x8B	PRI-PPS-MPS frame sent or receive
0x8C	PRI-PPS-EOM frame sent or receive
0x8D	PRI-PPS-EOP frame sent or receive
0x8E	EOR-NULL frame sent or receive
0x8F	EOR-MPS frame sent or receive
0x90	EOR-EOM frame sent or receive
0x91	EOR-EOP frame sent or receive
0x92	PRI-EOR-MPS frame receive
0x93	PRI-EOR-EOM frame sent or receive
0x94	PRI-EOR-EOP frame sent or receive
0x95	CRC error reported

Debugging - Set Event Bit 0x80

I/O Events

0x1F5	STAT_IO_OPEN_FILE
0x1F6	STAT_IO_CLOSE_FILE
0x1F7	STAT_IO_READ_ERR
0x1F8	STAT_IO_WRT_ERR
0x1F9	STAT_IO_BAD_LINES

EFT I/O Events

0x1FA	STAT_IO_BO_READ
0x1FB	STAT_IO_B1_READ
0x1FC	STAT_IO_BO_WRITE
0x1FD	STAT_IO_B1_WRITE
0x1FE	STAT_IO_BO_READ_DN
0x1FF	STAT_IO_B1_READ_DN
0x200	STAT_IO_BO_WRITE_DN
0x201	STAT_IO_B1_WRITE_DN
0x202	STAT_IO_IFD_READ
0x203	STAT_IO_IFD_WRITE
0x204	STAT_IO_HDR_READ
0x205	STAT_IO_HDR_WRITE
0x206	STAT_IO_1ST_IMG_READ
0x207	STAT_IO_NTH_IMG_READ
0x208	STAT_IO_IMG_WRITE
0x209	STAT_IO_FXP_WRITE
0x20A	STAT_IO_IFD_READ_DN
0x20B	STAT_IO_IFD_WRITE_DN
0x20C	STAT_IO_HDR_READ_DN
0x20D	STAT_IO_HDR_WRITE_DN
0x20E	STAT_IO_1ST_IMG_READ_DN
0x20F	STAT_IO_NTH_IMG_READ_DN
0x210	STAT_IO_IMG_WRITE_DN
0x211	STAT_IO_FXP_WRITE_DN
0x212	STAT_IO_WRT_NO_BFR
0x213	STAT_IO_RD_NO_BFR
0x214	STAT_IO_IFD_INVALID
0x215	STAT_IO_CLOSE_CLEANUP
0x216	STAT_IO_ERASE_LAST_PG
0x217	STAT_IO_WRT_LST_IFD
0x218	STAT_IO_WRT_LST_IFD_DN
0x219	STAT_IO_TIFF_WRT_DONE
0x21A	STAT_IO_FLUSH_LST_PG
0x21B	STAT_IO_TIFF_RD_DONE
0x21C	STAT_IO_FOUND_START_PG
0x21D	STAT_IO_END_PG_REACHED
0x21F	IS_FILE_EVENT (x)

Informational Events

NSF Types

0x79 STAT_INFO_NSF
0x7A STAT_INFO_NSS
0x7B STAT_INFO_NSC

ID Frames

0x7C STAT_INFO_CSI
0x7D STAT_INFO_TSI
0x7E STAT_INFO_CIG

Miscellaneous Information Frames

0x7F STAT_INFO_SEP
0x80 STAT_INFO_SUB
0x81 STAT_INFO_PWD

Negotiation Frames

0x82 STAT_INFO_DIS
0x83 STAT_INFO_DCS
0x84 STAT_INFO_DTC
0x85 STAT_INFO_CTC

ECM Information Frames

0x86 STAT_INFO_PPR

0x87 STAT_INFO_PPS_NULL
0x88 STAT_INFO_PPS_MPS
0x89 STAT_INFO_PPS_EOM
0x8A STAT_INFO_PPS_EOP

0x8B STAT_INFO_PRI_PPS_MPS
0x8C STAT_INFO_PRI_PPS_EOM
0x8D STAT_INFO_PRI_PPS_EOP

0x8E STAT_INFO_EOR_NULL
0x8F STAT_INFO_EOR_MPS
0x90 STAT_INFO_EOR_EOM
0x91 STAT_INFO_EOR_EOP

0x92 STAT_INFO_EOR_PPS_MPS
0x93 STAT_INFO_EOR_PPS_EOM
0x94 STAT_INFO_EOR_PPS_EOP
0x95 STAT_INFO_EOR_FCS

For Technical Support Only - Internal FAX Processing Events

Debugging - Set Event Bit 0x80

Call Back Event Codes for Frames without Data

0x1C8	STAT_FRM_BASE
0x1C9	STAT_FRM_CFR
0x1CA	STAT_FRM_CRP
0x1CB	STAT_FRM_CTR
0x1CC	STAT_FRM_DCN
0x1CD	STAT_FRM_EOM
0x1CE	STAT_FRM_EOP
0x1CF	STAT_FRM_EOR
0x1D0	STAT_FRM_ERR
0x1D1	STAT_FRM_FTT
0x1D2	STAT_FRM_MCF
0x1D3	STAT_FRM_MPS
0x1D4	STAT_FRM_PIN
0x1D5	STAT_FRM_PIP
0x1D6	STAT_FRM_PRI_EOM
0x1D7	STAT_FRM_PRI_EOP
0x1D8	STAT_FRM_PRI_MPS
0x1D9	STAT_FRM_RR
0x1DA	STAT_FRM_RNR
0x1DB	STAT_FRM_RTN
0x1DC	STAT_FRM_RTP

DIS/DCS/DTC Negotiation Problems Error Events

Note: The following events are generated as a result of problems in negotiation.

DIS Format Problems

0x44D	STAT_DIS_INVLD_RATE (DIS parameters mismatch on rates)
0x44E	STAT_DIS_INVLD_REC_LEN (DIS invalid record length)

DCS Format Problems

0x44F	STAT_DCS_INVLD_RATE (DCS invalid rate value)
0x450	STAT_DCS_INVLD_PAGESIZE (DCS invalid page size value)
0x451	STAT_DCS_INVLD_REC_LEN (DCS invalid record length)

Problems Computing DCS from DIS File Format and Parameters

0x452	STAT_NEG_Resolution
-------	---------------------

Received DCS did not match DIS capabilities for the following:

0x453	STAT_DCS_BAD_RATE (DCS and DIS mismatch on rate modems)
0x454	STAT_DCS_BAD_ENCODING (DCS and DIS mismatch on encoding)
0x455	STAT_DCS_BAD_ECM (DCS and DIS mismatch on ECM)
0x456	STAT_DCS_BAD_PAGESIZE (DCS and DIS mismatch page size)
0x457	STAT_DCS_BAD_200_RES (DCS and DIS mismatch on 200 x 200/204 x 391)
0x458	STAT_DCS_BAD_SPR_RES (DCS and DIS mismatch on SPR-RES)
0x459	STAT_DCS_BAD_300_RES (DCS and DIS mismatch on 300 x 300)
0x45A	STAT_DCS_BAD_400_RES (DCS and DIS mismatch on 400 x 400/408 x 391)
0x45B	STAT_NEG_NO_ID (ID required but not present)
0x45C	STAT_NEG_RAISE_RATE (Negotiate higher rate)
0x45D	STAT_NEG_LOWER_RATE (Negotiate lower rate)
0x45E	STAT_NEG_END_OF_RATES (No additional rates exist)

0x45F	STAT_NEG_INVLD_RATE
0x460	STAT_NEG_V27_2400 (Selected rate)
0x461	STAT_NEG_V27_4800
0x462	STAT_NEG_V27_7200
0x463	STAT_NEG_V27_9600
0x464	STAT_NEG_V17_7200
0x465	STAT_NEG_V17_9600
0x466	STAT_NEG_V17_12000
0x467	STAT_NEG_V17_14400
0x468	STAT_NEG_MH
0x469	STAT_NEG_MR
0x46A	STAT_NEG_MMR
0x46B	STAT_NEG_ECM
0x46C	STAT_NEG_RES_204 x 98
0x46D	STAT_NEG_RES_204 x 196
0x46E	STAT_NEG_RES_204 x 391
0x46F	STAT_NEG_RES_408 x 3918
0x470	STAT_NEG_RES_200 x 200
0x471	STAT_NEG_RES_300 x 300
0x472	STAT_NEG_RES_400 x 400
0x473	STAT_NEG_RES_600 x 600
0x474	STAT_NEG_A4
0x475	STAT_NEG_A3
0x476	STAT_NEG_B4
0x477	STAT_NEG_LETTER
0x478	STAT_NEG_LEGAL

Hardware Events

0x51	STAT_HW_V21_RX_START
0x52	STAT_HW_V21_RX_END
0x53	STAT_HW_V21_TX_START
0x54	STAT_HW_V21_TX_END
0x55	STAT_HW_RX_TRAIN_START
0x56	STAT_HW_RX_TRAIN_END
0x57	STAT_HW_RX_TRAIN_FAIL
0x58	STAT_HW_TX_TRAIN_START
0x59	STAT_HW_TX_TRAIN_END
0x5A	STAT_HW_HS_TX_START
0x5B	STAT_HW_HS_TX_END
0x5C	STAT_HW_HS_RX_START
0x5D	STAT_HW_HS_RX_END
0x5E	STAT_HW_V21_RX_FRAME
0x5F	STAT_HW_V21_TX_FRAME
0x60	STAT_HW_BUFUPDONE
0x61	STAT_HW_BUFNDONE
0x62	STAT_HW_V21_DETECTED
0x63	STAT_HW_HS_TX_READY
0x64	STAT_HW_HS_RX_READY
0x65	STAT_HW_PACKET_FAIL
0x66	STAT_HW_FAXSTOPPED

Call Processing Event 0x002E

	<p>0x67 STAT_HW_REMOTEDISCONNECT 0x68 STAT_HW_MODEMACTIVE 0x69 STAT_FRM_MCF</p>
	<p>0x42-0x48 (Positive Voice Detection/Answering Machine Detection (PVD/AMD)) Data[0-3] Calendar Time (seconds since Jan. 1, 1970) Data[4, 5] Milliseconds Remainder Data[6, 7] Milliseconds Since Last Event Data[8, 9] Frequency Band Detected (Events 0x47 and 0x48 only)</p> <p>0x50 DSP Series 2 FSK Events Data[0, 1] FSK Event Code 0x0001 FSK Transmit Complete 0x0002 FSK Transmit Failure 0x0003 FSK Init FSK Failure 0x0004 FSK Send SMS DLL Failure 0x0005 FSK Receive SMS DLL Failure 0x0006 FSK SMS DLL Released 0x0007 FSK Received Failed 0x0008 FSK Received Complete</p> <p>Data[2-5] FSK Event Data 0x00000001 Internal TLV Data Error 0x00000002 Timeout waiting for INIT ACK from DSP Chip 0x00000003 Invalid Data Type 0x00000004 Timeout waiting for FSK Message Sent Indication from DSP Chip 0x00000005 Timeout waiting for XMT DT-AS ACK from DSP Chip 0x00000006 Timeout waiting for XMT SAS ACK from DSP Chip 0x00000007 Timeout waiting for TE-ACK from TE 0x00000008 Timeout waiting for Incoming FSK DLL MSG from TE 0x00000009 Timeout waiting for Stop FSK ACK from DSP Chip 0x0000000A Timeout waiting for Host Response to Send Outgoing DLL 0x0000000B Invalid CAS Acknowledge (ACK) 0x0000000C Timeout waiting for Incoming DTMF Digit from CPE</p>
Checksum	

Call Progress Analysis Class Configure 0x00B3

SwitchKit Name CPAClassConfig

Type EXS API and SwitchKit API message

Description **Call Progress Analysis Class Configure 0x00B3**

Use this message to add a pattern, delete a pattern, or to change the parameters of a pattern in a call progress analysis class.

Sent by Host

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00B3)	3, 4	Message Type (0x00B3)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status MSB, LSB
		10	Checksum
8	Update All Flag (see also text below this table)		
9	Class ID		
10	Action		
11	Data[0]		
:	:		
:	Data[n]		
:	Checksum		

SwitchKit Code Configuration

```
CPAClassConfig (
    Node = integer,
    UpdateFlag = integer,
    Class = integer,
    Action = integer,
    Data = byte array);
```

C Structure

```
typedef struct {
    UBYTE UpdateFlag;
    UBYTE Class;
    UBYTE Action;
```

```
    UBYTE Data[250];  
} XL_CPAClassConfig;
```

C++ Class

```
class XLC_CPAClassConfig : public XLC_OutboundMessage {  
public:  
    UBYTE getUpdateFlag() const;  
    void setUpdateFlag(UBYTE x);  
    UBYTE getClass() const;  
    void setClass(UBYTE x);  
    UBYTE getAction() const;  
    void setAction(UBYTE x);  
    const UBYTE *getData() const;  
    UBYTE *getData();  
    void setData(UBYTE *x);  
};
```

**EXS API Hex Format -
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1	Length, MSB	1	Length, MSB (0x00)
2	Length, LSB (N)	2	Length, LSB (0x07)
3	Message Type, MSB (0x00)	3	Message Type, MSB (0x00)
4	Message Type, LSB (0xB3)	4	Message Type, LSB (0xB3)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status MSB, LSB
		10	Checksum
8	<p>Update All Flag (see also text below this table)</p> <p>0x00 Store new configuration but do not use it. Use if you are sending more configuration messages. Use for a new configuration that changes many parameters. The CSP does not use the new configuration until the last message has 0x01 in this field.</p> <p>0x01 Use new configuration immediately Use in the last configuration message. When the CSP detects this value, it uses this and all stored configuration commands to define a new call progress analysis class configuration.</p> <p>This field indicates whether you are going to update all fields with the configuration that you are defining in this and previous messages.</p> <p>For example, suppose you are updating several characteristics of call progress analysis. To do so, you are sending several of these <i>CPA Class Configure</i> messages. To prevent the CSP from performing a call progress analysis with a half-old and half-new configuration, you store the new configuration commands until the last new command is sent to the CSP.</p> <p>The <i>Call Progress Analysis Configuration Query</i> message reports configuration changes even when the <i>Update All Flag</i> is not set. But detection does not take place until the flag is set to 0x01.</p>		
9	<p>Class ID</p> <p>This field indicates the class number, which contains the information for detecting one group of patterns.</p>		
10	<p>Action</p> <p>0x01 Add member pattern 0x02 Delete member pattern 0x03 Change class parameter</p>		

11 . . .	<p>Data[0-n] This field or set of fields provides the necessary information to perform the action indicated in the <i>Action</i> field.</p> <p>0x01 Add Member Pattern Data[0] Pattern ID</p> <p>0x02 Delete Member Pattern Data[0] Pattern ID</p> <p>0x03 Change Class Parameter For parameter definitions, refer to the appropriate Developer's Guide. Data[0] Parameter</p> <p>0x01 Mode*</p> <p>0x02 Start Delay**</p> <p>0x03 Continuous Silence Timeout**</p> <p>0x04 Time to Wait for Pattern Confirmation**</p> <p>0x05 Time to Wait after Confirmation and No Report**</p> <p>0x06 Minimum Frequency Glitch Time*</p> <p>0x07 Minimum Silence Glitch Time**</p> <p>0x08 Maximum On Timer**</p> <p>0x09 Maximum Off Timer**</p> <p>0x0A Advanced Answer Detect***</p> <p>0x0B Mode Specific 1†</p> <p>0x0C Mode Specific 2‡</p> <p>*If Data[0] is 0x01, Data[1–2] are as follows: Data[1] Reserved Data[2] Mode This field is a bit mask that defines the mode. (0 = disabled, 1 = enabled). To enable multiple options, 'or' the bits together. Bit 0 Advanced Answer Detect Bit 1 Energy Detection Bits 2–7 Reserved</p> <p>**If Data[0] is 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, or 0x09, Data[1–2] are as follows: Data[1] The time in 10-ms units, MSB Data[2] The time in 10-ms units, LSB</p> <p>***If Data[0] is 0x0A, Data[1–2] are as follows: Data[1] 0x00 Data[2] Number of Frequency Bursts in 500 ms before answer is declared</p>
----------	--

	<p>‡If Data[0] is 0x0B, Data[1–2] are as follows:</p> <p>Data[1] 0x00 (Reserved)</p> <p>Data[2] Sensitivity Level (for more information on Energy Detection mode, refer to the DSP Series 2 Card Product Description in the Developer’s Guide Overview).</p> <p style="padding-left: 40px;">0x00 0 dBm</p> <p style="padding-left: 40px;">0x01 5 dBm</p> <p style="padding-left: 40px;">0x02 10 dBm</p> <p style="padding-left: 40px;">0x03 15 dBm</p> <p style="padding-left: 40px;">0x04 20 dBm</p> <p style="padding-left: 40px;">0x05 25 dBm</p> <p style="padding-left: 40px;">0x06 30 dBm</p> <p>‡If Data[0] is 0x0C, Data[1–2] are as follows:</p> <p>Data[1] Scan Duration in 20-ms increments, MSB</p> <p>Data[2] Scan Duration in 20-ms increments, LSB</p> <p>The value must be expressed in 20 ms increments, using 10 ms units. For example, you should use 0x0002 = 20 ms, 0x0004 = 40 ms, and so on.</p> <p>The CSP rounds down odd values. For example, the system rounds down 0x0003 to = 20ms. For more information on Energy Detection mode, refer to the appropriate Developer’s Guide.</p>
:	Checksum

Example 1 The table that follows shows a message that changes the number of frequency bursts per 500 ms (before declaring answer) to “5.”

MESSAGE		
Byte	Field	Field Value
0	Frame	0xFE
1	Length, MSB	0x00
2	Length, LSB	0x0B
3	Message Type, MSB	0x00
4	Message Type, LSB	0xB3
5	Reserved	0x00
6	Sequence Number	0x01
7	Logical Node ID	0xFF
8	Update All Flag	0x01
9	Class ID	0x00
10	Action	0x03 (Change Class Parameter)
11	Data[0] Parameter	0x0A (Advanced Answer Detect)
12	Data[1] Number of Frequency Bursts, MSB	0x00
13	Data[2] Number of Frequency Bursts, LSB	0x05 (5 bursts per 500 ms)
14	Checksum	0xD1

Example 2 This example shows the message sequence you would use to change the mode of Call Progress Analysis Class 3 to Energy Detection, with a sensitivity level of -30 dBm and a scan duration of 60ms.

To change the mode, you send the following three *Call Progress Analysis Class Configure* messages in this sequence:

Byte	Message Field	Field Value
8	Action	0x03 (Change Class Parameter)
9	Data[0] – Parameter	0x01 (Mode)
10	Data[1] – Reserved	0x00
11	Data[2] – Mode	0x02 (Energy Detection enabled)

Byte	Message Field	Field Value
8	Action	0x03 (Change Class Parameter)
9	Data[0] – Parameter	0x0B (Mode Specific 1)
10	Data[1] – Reserved	0x00
11	Data[2] – Sensitivity Level	0x06 (-30 dBm)

Byte	Message Field	Field Value
8	Action	0x03 (Change Class Parameter)
9	Data[0] – Parameter	0x0C (Mode Specific 2)
10	Data[1] – Reserved	0x00
11	Data[2] – Scan Duration	0x06 (60 ms)

Call Progress Analysis Configuration Query 0x008A

SwitchKit Name	CPAConfigQuery
Type	EXS API and SwitchKit API message
Description	Call Progress Analysis Configuration Query 0x008A Use this message to query call progress analysis (CPA) configuration information.
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {  
    UBYTE QueryEntity;  
    UBYTE DataType;  
} XL_CPAConfigQuery;
```

C Structure Response

```
typedef struct {  
    unsigned short Status;  
    UBYTE Data[251];  
} XL_CPAConfigQueryAck;
```

C++ Class

```
class XLC_CPAConfigQuery : public XLC_OutboundMessage {  
public:  
    UBYTE getQueryEntity() const;  
    void setQueryEntity(UBYTE x);  
    UBYTE getDataType() const;  
    void setDataType(UBYTE x);  
};
```

C++ Class Response

```
class XLC_CPAConfigQueryAck : public  
    XLC_AcknowledgeMessage {  
public:  
    unsigned short getStatus() const;  
    void setStatus(unsigned short x);  
    const UBYTE *getData() const;  
    UBYTE *getData();  
    void setData(UBYTE *x);  
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x0007)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x008A)	3, 4	Message Type (0x008A)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Query Entity This field indicates the characteristic of call progress analysis (CPA) about which the host is seeking information: 0x01 CPA Tone Group 0x02 CPA Pattern List 0x03 CPA Class	8, 9	Status MSB, LSB
9	Data[0] This field provides the necessary information to perform the action indicated in the <i>Query Entity</i> field. 0x01 CPA Tone Group Data[0] Tone Group ID 0x02 CPA Pattern List Data[0] Pattern ID 0x03 CPA Class Data[0] Class ID	10 ...	Data[0-N] This set of Data fields represents the CPA information requested by the host. The type and amount of data in the response depends on the value of the <i>Query Entity</i> field in the message. NOTE: Values in data fields marked with an asterisk (*) represent the number of 10ms units. 0x01 CPA Tone Group Data[0] Number of tones in group <u>Tone Descriptor 1</u> Data[1] Tone ID Data[2] Number of frequencies in tone Data[3, 4] Frequency Data[5, 6] Dbm Data[7, 8] Frequency (not included if Data[2] is 1) Data[9, 10] Dbm (not included if Data[2] is 1) Response continued below...
10	Checksum		
10 (Cont.)	: <u>Tone Descriptor N</u> Data[M] Tone ID Data[M+1] Number of frequencies in tone Data[M+2, M+3] Frequency Data[M+4, M+5] Dbm Data[M+6, M+7] Frequency (not included if Data[M+1] is 1) Data[M+8, M+9] Dbm (not included if Data[M+1] is 1)		

0x02	<p>CPA Pattern List</p> <p>Values in data fields marked with an asterisk (*) represent the number of 10ms units.</p> <p>Data[0] Reserved</p> <p>Data[1] Reserved</p> <p>Data[2] Pattern ID (Value returned upon detection)</p> <p>Data[3] Tone Group ID</p> <p>Data[4] Pattern Configuration</p> <p style="padding-left: 40px;">0x01 Last Interval Continuous</p> <p style="padding-left: 40px;">0x02 Detect Tone After Determination</p> <p style="padding-left: 40px;">0x04 Use as Internal Dialtone</p> <p>Data[5] CPA Result on Pattern Loss (for example, Answer)</p> <p>Data[6] Interval Cycles to Match</p> <p>Data[7] Interval Cycles to Report</p> <p>Data[8] Reserved</p> <p>Data[9] Interval Descriptor Count</p> <p><u>Interval Descriptor 1</u></p> <p>Data[10] Tone ID (0x00 for silence)</p> <p>Data[11] Reserved (0x00)</p> <p>Data[12,13] Minimum filter *</p> <p>Data[14,15] Maximum filter *</p> <p style="text-align: center;">:</p> <p><u>Interval Descriptor 8</u> (All 8 are sent even if not in use)</p> <p>Data[N] Tone ID (0x00 for silence)</p> <p>Data[N+1] Reserved (0x00)</p> <p>Data[N+2, N+3] Minimum filter *</p> <p>Data[N+4, N+5] Maximum filter *</p>
0x03	<p>CPA Class</p> <p>Values in data fields marked with an asterisk (*) represent the number of 10ms units.</p> <p>Data[0] Mode</p> <p>Data[1] Advanced Answer Detect</p> <p>Data[2,3] Start Delay</p> <p>Data[4,5] Continuous Silence Timeout *</p> <p>Data[6,7] Time to wait for pattern confirmation *</p> <p>Data[8,9] Time to wait after confirmation and no report *</p> <p>Data[10,11] Minimum frequency glitch time *</p> <p>Data[12,13] Minimum silence glitch time *</p> <p>Data[14,15] Maximum on timer *</p> <p>Data[16,17] Maximum off timer *</p> <p>Data[18,19] Mode Specific 1</p> <p>Data[20,21] Mode Specific 2</p> <p>Data[22] Number of patterns in class</p> <p>Data[23] Pattern ID (First one)</p> <p style="text-align: center;">:</p> <p>Data[N] Pattern ID (Last one)</p>
:	Checksum

Call Progress Analysis Pattern Configure 0x00B2

SwitchKit Name CPAPatternConfig

Type EXS API and SwitchKit API message

Description **Call Progress Analysis Pattern Configure 0x00B2**

Use this message to add, delete, or change a call progress analysis (CPA) pattern.

Sent by Host

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00B2)	3, 4	Message Type (0x00B2)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status MSB, LSB
		10	Checksum
8	Update All Flag		
9	Pattern ID		
10	Action		
11	Data [0]		
:	Data [n]		
:	Checksum		

SwitchKit Code **Configuration**

```
CPAPatternConfig (  
    Node = integer,  
    UpdateFlag = integer,  
    Pattern = integer,  
    Action = integer,  
    Data = byte array);
```

C Structure

```
typedef struct {  
    UBYTE UpdateFlag;
```

```
    UBYTE Pattern;  
    UBYTE Action;  
    UBYTE Data[250];  
} XL_CPAPatternConfig;
```

C++ Class

```
class XLC_CPAPatternConfig : public XLC_OutboundMessage {  
public:  
    UBYTE getUpdateFlag() const;  
    void setUpdateFlag(UBYTE x);  
    UBYTE getPattern() const;  
    void setPattern(UBYTE x);  
    UBYTE getAction() const;  
    void setAction(UBYTE x);  
    const UBYTE *getData() const;  
    UBYTE *getData();  
    void setData(UBYTE *x);  
};
```

**EXS API Hex Format -
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1	Length, MSB	1	Length, MSB (0x00)
2	Length, LSB (N)	2	Length, LSB (0x07)
3	Message Type, MSB (0x00)	3	Message Type, MSB (0x00)
4	Message Type, LSB (0xB2)	4	Message Type, LSB (0xB2)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status MSB, LSB
		10	Checksum
8	<p>Update All Flag</p> <p>0x00 Store new configuration but do not use it. Use this value if you are sending more configuration messages. You use this value for a new configuration that changes many parameters. The system does not use the new configuration until the last message has 0x01 in this field.</p> <p>0x01 Use new configuration immediately. Use this value in the last configuration message. When the system detects this value, it will use this and all stored configuration commands to define a new call progress analysis class configuration.</p> <p>This field indicates whether or not you are going to update all fields with the configuration you are defining in this and previous messages. For example, suppose you are updating several characteristics of call progress analysis. To do so, you send several of these <i>Call Progress Analysis Pattern Configure</i> messages. To prevent the CSP from performing a call progress analysis with a half-old and half-new configuration, you store the new configuration commands until the last new command is sent to the CSP.</p> <p>The <i>Call Progress Analysis Configuration Query</i> message reports configuration changes even when the <i>Update All Flag</i> is not set; however, detection does not happen until the flag is set to 0x01.</p>		

Call Progress Analysis Pattern Configure
0x00B2

9	<p>Pattern ID</p> <p>The number of the pattern to add, delete, or change.</p> <p>The following are the default IDs. Because you can modify the patterns, your pattern IDs may not correspond to this list.</p> <ul style="list-style-type: none"> 0x01 Ringback 0x02 Double Ringback 0x03 Busy 0x04 Reorder 0x05 PBX Intercept 0x06 SIT Intercept A 0x07 Vacant Code 0x08 Reorder-SIT 0x09 No Circuit LEC 0x0A Reorder-Carrier 0x0B No Circuit-Carrier 0x0C PBX Dialtone 0x0D Standard Dialtone 0x0E CPC Detection 0x0F PILOT 2000 ON 0x10 PILOT 2000 REMOVED 0x11 PILOT 1780 ON 0x12 SS7 PILOT 1780 REMOVED 0x13 FAX 0x14 SIT Intercept B
10	<p>Action</p> <ul style="list-style-type: none"> 0x01 Add/Replace Pattern 0x02 Delete Pattern 0x03 Change Pattern Parameter
11 . . .	<p>Data[0-N]</p> <p>This set of data fields provides the necessary information to perform the action indicated in the <i>Action</i> field. Valid entries for this field are listed below by <i>Action</i> field value.</p> <p>0x01 Add/Replace Pattern</p> <ul style="list-style-type: none"> Data[0] Pattern ID (Value to be returned upon detection) Data[1] Tone Group ID Data[2] Pattern Configuration <ul style="list-style-type: none"> This data byte is a bit mask you use to set the configuration options of a pattern. You can set multiple bits. <u>Bit</u> 0 Last Interval Continuous 1 Detect Tone After Determination 2 Use as Internal Dialtone 3 Receiver does not terminate after reporting this pattern. 4–7 Reserved Data[3] CPA Report on Pattern Loss (for example, Answer) Data[4] Interval Cycles to Match Data[5] Interval Cycles to Report Data[6] Reserved Data[7] Interval Descriptor Count (maximum is 8)

Call Progress Analysis Pattern Configure
0x00B2

	<p><u>Interval Descriptor 1</u> Data[8] Tone ID (0x00 for silence) Data[9] Reserved (0x00) Data[10,11] Minimum filter (given in 10ms units) Data[12,13] Maximum filter (given in 10ms units) :</p> <p><u>Interval Descriptor N</u> Data[N] Tone ID (0x00 for silence) Data[N+1] Reserved (0x00) Data[N+2, N+3] Minimum filter Data[N+4, N+5] Maximum filter</p> <p>0x02 Delete Pattern Data[0] Pattern ID to Delete</p> <p>0x03 Change Pattern Parameter Data[0] Parameter to Change 0x01 Pattern ID 0x02 Tone Group ID 0x03 Pattern Configuration 0x04 CPA Result on Pattern Loss (for example, Answer) 0x05 Interval Cycles to Match 0x06 Interval Cycles to Report 0x07 Reserved 0x08 Interval Data</p> <p>for all Parameters except 0x03 and 0x08 Data[1] New Parameter Value (</p> <p>for Parameter 0x03 Data[1] New Pattern Configuration This data byte is a bit mask you use to set the configuration options of a pattern. You can set multiple bits. <u>Bit</u> 0 Last Interval Continuous 1 Detect Tone After Determination 2 Use as Internal Dialtone 3-7 Reserved</p> <p>for Parameter 0x08 Data[1] Interval Descriptor Count Data[2] Interval Index Data[3] Tone ID Data[4] Reserved (0x00) Data[5-6] Minimum Filter (given in 10ms units) Data[7-8] Maximum Filter (given in 10ms units)</p>
:	Checksum

Example Message This example shows changing the “Interval Sequences to detect before reporting pattern” of Ringback to “2.”

Byte	Message	Values
0	Message Frame	0xFE
1	Length, MSB	0x00
2	Length, LSB	0x0A
3	Message Type, MSB	0x00
4	Message Type, LSB	0xB2
5	Reserved	0x00
6	Sequence Number	0x01
7	Logical Node ID	0xFF
8	Update All Flag	0x01
9	Pattern ID	0x01 (Ringback)
10	Action	0x03 (Change Parameter)
11	Data[0] – Parameter	0x06 (Interval Cycles to Report)
12	Data[1] – Value	0x02
13	Checksum	0xC1

Call Progress Analysis Result 0x0034

SwitchKit Name CPAResult

Type EXS API and SwitchKit API message

Description **Call Progress Analysis Result 0x0034**

The CSP sends this message to report the result of Call Progress Analysis on a channel. Only one result is reported per message.

Sent by CSP

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short Span;
    UBYTE Channel;
    UBYTE Result;
    UBYTE Diag1;
    UBYTE Diag2;
    UBYTE Diag3;
} XL_CPAResult;
```

C++ Class

```
class XLC_CPAResult : public XLC_OneChannelMessage {
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getResult() const;
    void setResult(UBYTE x);
    UBYTE getDiag1() const;
    void setDiag1(UBYTE x);
    UBYTE getDiag2() const;
    void setDiag2(UBYTE x);
    UBYTE getDiag3() const;
    void setDiag3(UBYTE x);
};
```

Resent in EXS API This message is resent once after five seconds.

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0005)
3, 4	Message Type (0x0034)	3, 4	Message Type (0x0034)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u> Address Method 0x00 - Individual AEs <hr/> Number of AEs to follow <hr/> AE 0x0D Channel	8	Checksum
:	Call Progress Analysis Result 0x01 Ringback * 0x02 Double Ringback 0x03 Busy 0x04 Reorder 0x05 PBX Intercept 0x06 SIT Intercept A 0x07 Vacant Code 0x08 Reorder-LEC 0x09 No Circuit-LEC 0x0A Reorder-Carrier 0x0B No Circuit-Carrier 0x0C PBX Dialtone 0x0D Standard Dialtone 0x13 Fax Tone Detected 0x14 SIT Intercept B 0x80 Answer 0x81 Reserved 0x82 Continuously On Tone 0x83 Initial Silence Timeout 0x84 Not Determined Timeout 0x85 Determined Timeout 0x86 Answer determined through line signaling 0x87 Reserved * Ringback is detected on the fourth cycle of ringback/silence		
:	Reserved (0x00)		
:	Reserved (0x00)		
:	Reserved (0x00)		
:	Checksum		

CancelUserTimer

Type:	SwitchKit API message
Description	<p>Use this message to cancel a previously sent <i>UserTimer</i> message or group of <i>UserTimer</i> messages. This will cause the canceled timer to go off with a negative status value, SK_CANCELED. All handler functions which are installed to handle <i>UserTimer</i> expirations should be prepared to do nothing if the Status is not positive. If you want to cancel a <i>UserTimer</i>, the GroupTag should be set when the <i>UserTimer</i> is sent. This GroupTag should be saved for later use.</p> <p>When you want to cancel that timer, you need to send a <i>CancelUserTimer</i> message with the appropriate GroupTag entered. All currently unexpired <i>UserTimer</i> messages from that application with the corresponding GroupTag are canceled, and the acknowledgment sent back will contain the number of timers that were canceled. Depending on how the <i>UserTimer</i> messages are set up, you can cancel a set of <i>UserTimer</i> messages with a single <i>CancelUserTimer</i> message.</p> <p>It is important to realize that the GroupTag domain is unique to each application. This prevents one application from canceling another application's <i>UserTimer</i>. If multiple applications use the same GroupTag in a timer, and one application cancels that tag, only the timers from that application will be canceled. All other timers will still expire normally.</p>
Sent by	Application
Arguments	The following table shows the arguments that you can modify:

Argument	Description
GroupTag	<p>Cancellation of a timer requires that this field be set at creation of the timer and then remembered. When you want to cancel that timer, you need to send a <i>CancelUserTimer</i> message with the appropriate GroupTag entered. All currently unexpired UserTimer messages from that application with the corresponding GroupTag are canceled.</p> <p>It is important to realize that the GroupTag domain is unique to each application. This prevents one application from canceling another application's <i>UserTimer</i>. If multiple applications use the same GroupTag in a timer, and one application cancels that tag, only the timers from that application will be canceled. All other timers will still expire normally.</p>

C Structure typedef struct {
 int GroupTag;
 } SK_CancelUserTimer;

C Structure Response typedef struct {
 int Status;
 int GroupTag;
 int NumCanceled;
 } SK_CancelUserTimerAck;

C++ Class class SKC_CancelUserTimer : public SKC_ToolkitMessage {
 public:
 int getGroupTag() const;
 void setGroupTag(int x);
 };

C++ Class Response class SKC_CancelUserTimerAck : public SKC_ToolkitAck {
 public:
 int getStatus() const;
 void setStatus(int x);
 int getGroupTag() const;
 void setGroupTag(int x);
 int getNumCanceled() const;
 void setNumCanceled(int x);
 };

Card Population Query 0x0007

SwitchKit Name	CardPopulationQuery
Type	EXS API and SwitchKit API message
Description	<p>Card Population Query 0x0007</p> <p>Use this message to query the cards installed in the system. The response indicates the Card Type installed in each slot (or virtual slots) in the system.</p>
Related Information	<p>Chassis Slot Numbering</p> <p>Card Types</p>
Sent by	Host
SwitchKit Code	<p>C Structure</p> <pre>typedef struct { } XL_CardPopulationQuery;</pre> <p>C Structure Response</p> <pre>typedef struct { unsigned short Status; UBYTE Info[96]; } XL_CardPopulationQueryAck;</pre> <p>C++ Class</p> <pre>class XLC_CardPopulationQuery : public XLC_OutboundMessage { public: };</pre> <p>C++ Class Response</p> <pre>class XLC_CardPopulationQueryAck : public XLC_AcknowledgeMessage { public: unsigned short getStatus() const; void setStatus(unsigned short x); const UBYTE *getInfo() const; UBYTE *getInfo(); void setInfo(UBYTE *x); };</pre>
Overview of message	The following table provides an overview of this message. The table following it provides the detail for each byte.

Card Population Query 0x0007

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x0005)	1, 2	Length (0x0067)
3, 4	Message Type (0x0007)	3, 4	Message Type (0x0007)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Checksum	8, 9	Status (MSB, LSB)
		10	Slot 0* Card Type
		11	Slot 1* Card Type
		73	Card Type
		74-105	Slot 63 Card Type
		106	Checksum

**EXS API Hex Format -
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x0005)	1, 2	Length (0x0067)
3, 4	Message Type (0x0007)	3, 4	Message Type (0x0007)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Checksum	8, 9	Status MSB, LSB
		10	Slot 0* Card Type
		11	Slot 1* Card Type (Response continued below.)

Card Population Query 0x0007

:	Card Type
	0x00 8-Span ST1LC
	0x01 4-Span ST1LC
	0x03 MFDSP
	0x0A SS7
	0x0B SE1LC 4-Span
	0x0C SE1LC 8-Span
	0x0E SJ1LC 4-Span
	0x13 ISDN PRI
	0x14 DASS2
	0x15 DSP-ONE
	0x16 T-ONE 8-Span
	0x17 E-ONE 8-Span
	0x18 T-ONE 16-Span
	0x19 E-ONE 16-Span
	0x1D J-ONE 8-Span
	0x1E J-ONE 16-Span
	0x1F DS3
	0x33 Subrate Controller
	0x42 EXNET Connect PCI H.100
	0x50 SS7 PQ
	0x54 EXNET ONE
	0x60 VDAC
	0x61 VDAC VOIP Module
	0x65 IP Network Interface Series 2
	0x6B DSP Series 2
	0x6D IP Network Interface Series 2 VoIP Module
	0x71 SS7 Series 3
	0x72 CSP Matrix I/O
	0x73 CSP Matrix 2000
	0x75 CSP Matrix 1000
	0x76 ISDN Series 3
	0x79 IP Signaling Series 3
	0x80 Virtual 32 Span VDAC-ONE or IP Media Series 2
	0xCA Multi-Function Media I/O
	0xCC DS3 Standby I/O
	0xCF VDAC I/O
	0xD2 Virtual T1 16-Span
	0xD5 Virtual E1/J1 16-Span

	0xD8 CCS I/O Series 3 0xDB J-ONE Redundant I/O 0xDC J-ONE Standby I/O 0xDD DS3 Redundant I/O 0xDF CCS I/O 0xE0 E-ONE 75 Ohm Redundant I/O 0xE1 T-ONE 100 Ohm Redundant I/O 0xE2 E-ONE 120 Ohm Redundant I/O 0xE3 E-ONE 75 Ohm Standby I/O 0xE4 T-ONE 100 Ohm Standby I/O 0xE5 E-ONE 120 Ohm Standby I/O 0xE8 SS7 Multi-Protocol I/O 0xF0 Power Supply 0xF5 EXNET Ring I/O 0xF6 Rear Fan Tray 0xF7 SS7 Redundant I/O 0xF8 Front Fan Tray 0xF9 ISDN Redundant I/O 0xFA Midplane 0xFE Dead Card – All subsequent data fields are not applicable and all other bytes in the message should be ignored. 0xFF Empty Slot – All subsequent data fields are not applicable.
73	Slot 63 Card Type
74-105	Slot 64-95 Card Type - Virtual Slot Configuration (See Virtual Slots below.)
106	Checksum

***Slot Numbers**

Front of Chassis
0x00–0x0F Line Card Slots 0–15
0x00–0x03 Line Card Slots 0–3
0x10 Front Fan Tray
0x11 Power2/ PSC 2
0x12 Power1/PSC 1
0x16 Fan Tray
0x20 CSP Matrix Series 3 Card 1
0x21 CSP Matrix Series 3 Card 2
Back of Chassis
0x13 Midplane
0x14 CSP Matrix Card I/O for CPU1
0x15 CSP 2000 Matrix I/O for CPU2
0x16 Rear Fan Tray
0x30–0x3F Redundant Line Card I/Os
0x30–0x33 Redundant Line Card I/Os
Virtual
0x40-0x5F Virtual Slots

Card Status Query 0x0083

SwitchKit Name	CardStatusQuery
Type	EXS API and SwitchKit API message
Description	<p>Card Status Query 0x0083</p> <p>Use this message to get a report on a specified card type. The host can also send this message to the active or standby CSP Matrix Series 3 Card.</p>
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {
    } XL_CardStatusQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE AddrInfo[30];
    UBYTE CardType;
    UBYTE CardStatus;
    UBYTE ConfTestRslts;
    UBYTE ArtRevision;
    UBYTE FunctRevision;
    UBYTE ROMMajorRevision;
    UBYTE ROMMinorRevision;
    UBYTE RAMMajorRevision;
    UBYTE RAMMinorRevision;
    unsigned short SerialNumber;
    unsigned short RAMSize;
    UBYTE CardSpecifics[384];
    UBYTE HardwareConfig[16];
    UBYTE CardID;
    UBYTE RAMBuildNum;
    unsigned short ConfigTag;
    UBYTE LineCardSpeed;
    UBYTE SoftwareResetReason;
    } XL_CardStatusQueryAck;
```

C++ Class

```
class XLC_CardStatusQuery : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    // Extended addressing functions
    // Slot AIB functions
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
```

```
// Module AIB functions
UBYTE getModule() const;
void setModule(UBYTE x);
};
```

C++ Class Response

```
class XLC_CardStatusQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getModule() const;
    void setModule(UBYTE x);
    UBYTE getCardType() const;
    void setCardType(UBYTE x);
    UBYTE getCardStatus() const;
    void setCardStatus(UBYTE x);
    UBYTE getConfTestRslts() const;
    void setConfTestRslts(UBYTE x);
    UBYTE getArtRevision() const;
    void setArtRevision(UBYTE x);
    UBYTE getFunctRevision() const;
    void setFunctRevision(UBYTE x);
    UBYTE getROMMajorRevision() const;
    void setROMMajorRevision(UBYTE x);
    UBYTE getROMMinorRevision() const;
    void setROMMinorRevision(UBYTE x);
    UBYTE getRAMMajorRevision() const;
    void setRAMMajorRevision(UBYTE x);
    UBYTE getRAMMinorRevision() const;
    void setRAMMinorRevision(UBYTE x);
};
```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x0083)	3, 4	Message Type (0x0083)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB (starting with byte 0)	8, 9	Status MSB, LSB
:	:		
:	Checksum	10	AIB

Card Status Query 0x0083

	Response continued below.
:	Card Type
:	Card Status
:	Confidence Test Results
:	PC Artwork Revision
:	Functional Revision
:	ROM Major Revision
:	ROM Minor Revision
:	RAM Major Revision
:	RAM Minor Revision
2 bytes	Serial Number MSB, LSB
2 bytes	RAM Size MSB, LSB
:	Card Information (Field length varies)
16 bytes	Hardware Configuration (16 bytes)
:	Card ID
:	RAM Build Number
2 bytes	Configuration Tag MSB, LSB
:	CPU Speed
:	Software Reset Reason
:	Checksum

**EXS API Hex Format -
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x0083)	3, 4	Message Type (0x0083)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs	8, 9	Status MSB, LSB
	Number of AEs to follow	:	AIB Same AIB sent in message
	AEs 0x01 Slot or 0x42 VoIP Module		
:	Checksum		
Response continued below.			
:	Card Type		
	0x00 8-Span ST1LC		
	0x01 4-Span ST1LC		
	0x03 MFDSP		
	0x0A SS7		
	0x0B SE1LC 4-Span		
	0x0C SE1LC 8-Span		
	0x0E SJ1LC 4-Span		
	0x13 ISDN PRI		
	0x14 DASS 2		
	0x15 DSP-ONE		
	0x16 T-ONE 8-Span		
	0x17 E-ONE 8-Span		
	0x18 T-ONE 16-Span		
	0x19 E-ONE 16-Span		
	0x1D J-ONE 8-Span		
	0x1E J-ONE 16-Span		
	0x1F DS3		
	0x33 Subrate Controller		

0x42	EXNET Connect PCI H.100
0x50	SS7 PQ
0x54	EXNET-ONE
0x60	VDAC ONE
0x61	VDAC VoIP Module
0x65	IP Network Interface Series 2
0x6B	DSP Series 2
0x6D	IP Network Interface Series 2 VoIP Module
0x6E	Mindspeed VoIP Module
0x71	SS7 Series 3
0x72	CSP 2000 Matrix I/O
0x73	CSP 2000 Matrix 2000
0x75	CSP 2000 Matrix 1000
0x76	ISDN Series 3
0x79	IP Signaling Series 3
0x80	Virtual 32 Span VDAC-ONE or IP Media Series 2
0x90	DSP Series 2 Plus
0x91	IP Network Interface Series 3
0x92	Multi-Function Media I/O Plus
0xAA	Illegal I/O
0xCA	Multi-Function Media I/O
0xCC	DS3 Standby I/O
0xCF	VDAC I/O
0xD2	Virtual T1 16-span
0xD5	Virtual E1/J1 16-Span
0xD8	CCS I/O Series 3
0xDB	J-ONE Redundant I/O
0xDC	J-ONE Standby I/O
0xDD	DS3 Redundant I/O
0xDF	CCS I/O
0xE0	E-ONE 75 Ohm Redundant I/O
0xE1	T-ONE 100 Ohm Redundant I/O
0xE2	E-ONE 120 Ohm Redundant I/O
0xE3	E-ONE 75 Ohm Standby I/O
0xE4	T-ONE 100 Ohm Standby I/O
0xE5	E-ONE 120 Ohm Standby I/O
0xE8	SS7 Multi-Protocol I/O
0xF0	Power Supply Card
0xF3	Matrix I/O
0xF5	EXNET Ring I/O
0xF6	Rear Fan Tray
0xF7	SS7 Redundant I/O
0xF8	Front Fan Tray
0xF9	ISDN Redundant I/O
0xFA	Midplane
0xFE	Dead Card: All subsequent data fields do not apply and all other bytes in the message should be ignored.
0xFF	Empty Slot: All subsequent data fields do not apply.

:	<p>Card Status This field is a bit mask that applies to the following cards:</p> <p>CSP Matrix Series 3 Card, T-ONE, E-ONE, J-ONE, DS3, EXNET-ONE, SS7-PQ, SS7 Series 3, DSP-ONE, MFDSP, DSP Series 2, DSP Series 2 Plus, ISDN PRI, ISDN Series 3, ST1LC, SE1LC, SJ1LC, VDAC-ONE, IP Network Interface Series 2/3, IP Signaling Series 3</p> <p>Bit</p> <p>0-1 Reset Reason 0 Dialogic use only 1 Power-up 2 Dialogic use only 3 Reset button pushed</p> <p>2-3 Card State 0 In service 1 The CSP has taken the card out of service 2 The host has taken the card out of service 3 The host has put the card in a standby state</p> <p>4 Invalid battery-backed configuration data. The host must reconfigure the card</p> <p>5 Faults are logged Faults may be obtained using a <i>Fault Log Query</i> message. Obtain a Fault Log and report it to Dialogic Technical Support as soon as possible.</p> <p>6 Failed confidence tests. Information about the confidence test which failed is in the "Confidence Test Results" field.</p> <p>7 General hardware failure The card is off-line and must be serviced. See the Hardware Configuration Field for additional information on the ISDN PRI card.</p>
---	---

	<p>Fan Tray</p> <p>Bit</p> <p>0 Reserved</p> <p>1 Fan 6 failed</p> <p>2 Fan 5 failed</p> <p>3 Fan 4 failed</p> <p>4 Fan 3 failed</p> <p>5 Fan 2 failed</p> <p>6 Fan 1 failed</p> <p>7 General hardware failure. The card is off-line and must be serviced.</p> <p>Power Supply Card</p> <p>Bit</p> <p>0 Module 3 failed</p> <p>1 Module 3 in tolerance</p> <p>2 Module 2 failed</p> <p>3 Module 2 in tolerance</p> <p>4 Module 1 failed</p> <p>5 Module 1 in tolerance</p> <p>6 Out of Tolerance</p> <p>7 General hardware failure. The card is off-line and must be serviced.</p> <p>DS3, T1, E1, and J1 Redundant and Standby I/O Cards</p> <p>Bit</p> <p>0-5 Reserved</p> <p>6 Output Enabled</p> <p>7 Relays Enabled (0=Normal, 1=Switchover)</p> <p>Matrix I/O Card</p> <p>Bit</p> <p>0 Interface B link state: 0=Down, 1=Up</p> <p>1 Interface A link state: 0=Down, 1=Up</p> <p>2 The active Ethernet Interface: 0=Interface A. 1=Interface B</p>
:	<p>Confidence Test Results</p> <p>This field is bit-defined. It applies to the CSP Matrix Series 3 Card cards and line cards.</p> <p>Bit</p> <p>0-5 Reserved</p> <p>6 Bit set indicates RAM test failure</p> <p>7 Bit set indicates ROM checksum failure</p>
:	<p>PC Artwork Revision</p> <p>An ASCII character in the range "A" to "Z" (0x41-0x5B) that indicates the revision level of the card's Printed Circuit Artwork.</p>
:	<p>Functional Revision</p> <p>The card's functional revision level</p>
:	<p>ROM Major Revision</p> <p>The major revision level of the firmware running on the card.</p>
:	<p>ROM Minor Revision</p> <p>The minor revision level of the firmware running on the card.</p>
:	<p>RAM Major Revision</p> <p>The major revision level of the system software running on the card.</p>
:	<p>RAM Minor Revision</p> <p>The minor revision level of the system software running on the card.</p>

:	<p>Serial Number MSB, LSB This two-byte field indicates the serial number of the card.</p>
:	<p>RAM Size MSB, LSB This two-byte field indicates the amount of RAM on the card, in 1K increments. For the VDAC VoIP Card and VDAC VoIP Module, RAM is measured in 1 MB increments.</p>
:	<p>Card Information (Field length varies) The card information, including the length of the field, varies per card type. The system uses the value 0x00 for fields that do not apply to the card or for all fields, if there is no card information.</p> <p><u>IP Signaling Series 3 Card</u> No card specific or hardware specific information is sent for this card.</p> <p>T-ONE, E-ONE, J-ONE, DS3, ST1LC, SE1LC, VDAC-ONE, IP Network Interface Series 2/3, and SS7 Multi- Protocol cards. The card information for these cards indicates the logical spans assigned to the card. The logical spans are indicated in the Logical Span AIB (0x0C) as shown below. The CSP uses the value 0xFF to indicate that no span has been assigned. An AIB for a 4-span line card has four address elements and the value 0x00 for the remaining bytes. An AIB for an 8-span line card has eight address elements and the value 0x00 for the remaining bytes.</p> <p>Data[0] Address Method (0x00, Single Entity) Data[1] Number of Address Elements (equal to number of spans)</p> <p>Address Element 1: Span 0 Data[2] Address Type (0x0C, Logical Span) Data[3] Data Length (0x02) Data[4] Logical Span ID, MSB Data[5] Logical Span ID, LSB</p> <p>Address Element N: Span N Data[...] Address Type (0x0C, Logical Span) Data[...] Data Length (0x02) Data[...] Logical Span ID, MSB Data[...] Logical Span ID, LSB</p>

MF DSP and DSP-ONE Cards

The card information for the DSP-ONE cards indicates the function types for DSPs on all SIMMs on the card.

NOTE: For more information on function types, refer to the *DSP SIMM Configure (0xC0)* message.

<u>Byte</u>	<u>Function Type</u>
Data[0]	SIMM 0/DSP 0
Data[1]	SIMM 0/DSP 1
Data[2]	SIMM 0/DSP 2
Data[3]	SIMM 0/DSP 3
Data[4]	SIMM 1/DSP 0
Data[5]	SIMM 1/DSP 1
Data[6]	SIMM 1/DSP 2
Data[7]	SIMM 1/DSP 3
Data[8]	SIMM 2/DSP 0
Data[9]	SIMM 2/DSP 1
Data[10]	SIMM 2/DSP 2
Data[11]	SIMM 2/DSP 3
Data[12]	SIMM 3/DSP 0
Data[13]	SIMM 3/DSP 1
Data[14]	SIMM 3/DSP 2
Data[15]	SIMM 3/DSP 3
Data[16–33]	Reserved

DSP Series 2 and DSP Series 2 Plus Cards

The card information for the DSP Series 2 and DSP Series 2 Plus cards indicates the function types for all DSPs on all Modules on the card, because each DSP can do multiple configurations.

<u>Byte</u>	<u>Function Type</u>
Data[0]	Module 0 / DSP 0 / RCV 0
Data[1]	Module 0 / DSP 0 / TXMT 0
Data[2]	Module 0 / DSP 0 / RCV 1
Data[3]	Module 0 / DSP 0 / TXMT 1
Data[4]	Module 0 / DSP 1 / RCV 0
Data[5]	Module 0 / DSP 1 / TXMT 0
Data[6]	Module 0 / DSP 1 / RCV 1
Data[7]	Module 0 / DSP 1 / TXMT 1
Data[8]	Module 0 / DSP 2 / RCV 0
Data[9]	Module 0 / DSP 2 / TXMT 0
Data[10]	Module 0 / DSP 2 / RCV 1
Data[11]	Module 0 / DSP 2 / TXMT 1
Data[12]	Module 0 / DSP 3 / RCV 0
Data[13]	Module 0 / DSP 3 / TXMT 0
Data[14]	Module 0 / DSP 3 / RCV 1
Data[15]	Module 0 / DSP 3 / TXMT 1
Data[16]	Module 1 / DSP 0 / RCV 0
Data[17]	Module 1 / DSP 0 / TXMT 0
Data[18]	Module 1 / DSP 0 / RCV 1
Data[19]	Module 1 / DSP 0 / TXMT 1
Data[20]	Module 1 / DSP 1 / RCV 0
Data[21]	Module 1 / DSP 1 / TXMT 0
Data[22]	Module 1 / DSP 1 / RCV 1
Data[23]	Module 1 / DSP 1 / TXMT 1
Data[24]	Module 1 / DSP 2 / RCV 0
Data[25]	Module 1 / DSP 2 / TXMT 0
Data[26]	Module 1 / DSP 2 / RCV 1
Data[27]	Module 1 / DSP 2 / TXMT 1
Data[28]	Module 1 / DSP 3 / RCV 0
Data[29]	Module 1 / DSP 3 / TXMT 0
Data[30]	Module 1 / DSP 3 / RCV 1
Data[31]	Module 1 / DSP 3 / TXMT 1
Data[32]	Reserved
Data[33]	Reserved

	<p><u>EXNET-ONE</u> This card generates no special information for this field. All bytes are 0x00.</p> <p><u>SS7 Card</u> Data[0] Maximum Configurable Links for card Data[1] Slot Number of Active SS7 Card (see Note) Data[2] Slot Number of Secondary SS7 Card in Redundant Pair (see Note) Data[3] Stack ID Data[4] Stack ID Data[5] Stack ID Data[6] Stack ID Data[7-15]Reserved</p> <p>NOTE: Do not try to determine the redundancy status from bytes Data[1] and Data[2]. To determine the redundancy status, use the <i>CCS Redundancy Query</i> message.</p> <p>Redundant I/O and Standby I/O A Slot AIB indicates the slot number of the other I/O (Standby I/O or Redundant I/O) involved in a switchover.</p> <p><u>PCI H.100 Card</u> The card information fields depend upon on H.100 bus clock speed and the number of spans per channel. 4MHz and 24 channels per span display 21 span entries 4MHz and 32 channels per span display 16 span entries 8MHz and 24 channels per span display 80 span entries 8MHz and 32 channels per span display 64 span entries</p>
:	<p>Hardware Configuration (16 bytes) This 16-byte field describes the hardware configuration of the card specified. This field applies to only the cards described below.</p> <p><u>DS3 Cards</u></p> <p>Data[0] Number of SS7 Spans Supported: 0x18 24 Spans 0x1C 28 Spans</p> <p>Data[1] DS3 Offset Frammer Status Bit 0 - DS3 Offset 0 Frammer Status 0 - Out of Service 1 - In Service (default) Bit [1-7] Reserved</p> <p>Data [2-13] Reserved</p> <p>Data[14] Hardware Switch 1 Settings (also refer to the Hardware Product Descriptions) (the right-most bit is dip 1)</p> <p>Data[15] Total Available Spans</p>

ST1LC and SE1LC cards

Data[0] Informs the host of primary and secondary loop timing clock sources configured on the card.
Bits 0-2 If secondary loop timing is enabled, this field contains the offset of the span for which it is enabled.
Bit 3 Secondary Loop timing
0 Disabled
1 Enabled
Bit 4-6 If primary loop timing is enabled, this field contains the offset of the span for which it is enabled.
Bit 7 Primary Loop timing
0 Disabled
1 Enabled
Data[1-15] Reserved

T-ONE, E-ONE, J-ONE cards

Data 0-3 Informs the host of primary and secondary loop timing clock sources configured on the card

Data[0] Primary Loop Timing
0 Not Enabled
1 Enabled
Data[1] If Primary Loop Timing is enabled, this field contains the span offset of the span for which it is enabled.
Data[2] Secondary Loop Timing
0 Not enabled
1 Enabled
Data[3] If Secondary Loop Timing is enabled, this field contains the span offset of the span for which it is enabled.
Data[4-13] Reserved
Data[14] Hardware Switch 1 Settings (also refer to the Hardware Product Descriptions) (the right-most bit is dip 1)
Data[15] Total Available Spans

MFDSP and DSP-ONE Card

This field informs the host of the number and types of DSP SIMMs installed on the card.

Data[0] Number of DSP SIMMs installed on the card

Data[1] Reserved

Data[2] SIMM 0 Type

0x00 C31 SIMM

0x01 VRA SIMM

0x03 No SIMM installed

Data[3] SIMM 1 Type

0x00 C31 SIMM

0x01 VRA SIMM

0x03 No SIMM installed

Data[4] SIMM 2 Type

0x00 C31 SIMM

0x01 VRA SIMM

0x03 No SIMM installed

Data[5] SIMM 3 Type

0x00 C31 SIMM

0x01 VRA SIMM

0x03 No SIMM installed

Data 6-9: SIMM State 0 - SIMM State 3

0x00 In Service

0x01 Out of Service (or No SIMM Installed)

Data[6] SIMM 0 State

Data[7] SIMM 1 State

Data[8] SIMM 2 State

Data[9] SIMM 3 State

Data[10-15]Reserved

DSP Series 2 and DSP Series 2 Plus Cards

This field informs the host of the number and types of DSP Modules installed on the card.

- Data[0] Number of Modules installed on the card
- Data[1] Reserved
- Data[2] Module 0 Type
 - 0x6C DSP 2 Module
 - 0x03 No Module installed
- Data[3] Module 1 Type
 - 0x6C DSP 2 Module
 - 0x03 No Module installed
- Data[4, 5] Reserved
- Data 6-13: DSP Chip States
 - 0x00 In Service
 - 0x01 Out of Service (or No Module Installed)
- Data[6] Module 0 / Chip 0 State
- Data[7] Module 0 / Chip 1 State
- Data[8] Module 0 / Chip 2 State
- Data[9] Module 0 / Chip 3 State
- Data[10] Module 1 / Chip 0 State
- Data[11] Module 1 / Chip 1 State
- Data[12] Module 1 / Chip 2 State
- Data[13] Module 1 / Chip 3 State
- Data[14-15] Reserved

ISDN PRI Card

- Data[0] Munich Chip Count
- Data[1-15] Reserved

EXNET-ONE card

Data[0–3]Informs host of primary and secondary loop timing clock sources configured from ring timing on the card. For EXNET-ONE cards configured as a Single Ring Redundancy Standby, the Loop Timing fields are always disabled.

- Data[0] Primary Loop Timing
 - 0x00 Not Enabled
 - 0x01 Enabled
- Data[1] Reserved
- Data[2] Secondary Loop Timing
 - 0x00 Not Enabled
 - 0x01 Enabled
- Data[3] Reserved

SS7 cards

The system does not report hardware configuration information for these cards. All bytes are 0x00.

SS7 Multi-Protocol I/O card

Data[0] Channel 0 Data Rate
0x00 64 Kbps
0x01 56 Kbps, for non-V.35 applications
0x80 V.35, 56 Kbps
0x81 48 Kbps, for non-V.35 applications
0xC0 V.35, 48 Kbps

Data[1] Channel 0 Electrical Interface
0x00 No Timing
0x01 V.35, DTE
0x02 V.35, DCE

Data[2] Channel 1 Data Rate (same possibilities as Channel 0, above)
Data[3] Channel 1 Electrical Interface (same possibilities as Channel 0, above)
Data[4] Channel 2 Data Rate (same possibilities as Channel 0, above)
Data[5] Channel 2 Electrical Interface (same possibilities as Channel 0, above)
Data[6] Channel 3 Data Rate (same possibilities as Channel 0, above)
Data[7] Channel 3 Electrical Interface (same possibilities as Channel 0, above)

VDAC-ONE card

Data[0] Number of VoIP Modules (1 - 4)
Data[1] Number of channels supported (MSB)
Data[2] Number of channels supported (LSB)
Data[3] Installed VoIP Modules (Bit Field):
 Bit 0 VoIP Module 0 (0 = not installed, 1 = installed)
 Bit 1 VoIP Module 1 (0 = not installed, 1 = installed)
 Bit 2 VoIP Module 2 (0 = not installed, 1 = installed)
 Bit 3 VoIP Module 3 (0 = not installed, 1 = installed)
Data[4] Module 0 state (0x05=ready, other values=not ready)
Data[5] Module 1 state (0x05=ready, other values=not ready)
Data[6] Module 2 state (0x05=ready, other values=not ready)
Data[7] Module 2 state (0x05=ready, other values=not ready)
Data[8] Port Consumption Mode
 1 Reserved
 2 Port-Consuming
Data[9-15] Reserved (should be zero)

VDAC VoIP Module

Data[0] Total number of channels supported on the module
Data[1] Number of channels supported for DSP 0
Data[2] Number of channels supported for DSP 1
Data[3] Number of channels supported for DSP 2
Data[4] Number of channels supported for DSP 3
Data[5] Number of channels supported for DSP 4
Data[6] Number of channels supported for DSP 5
Data[7] Number of channels supported for DSP 6
Data[8] Number of channels supported for DSP 7
Data[9-15]Reserved (should be zero)

IP Network Interface Series 2 card

Data[0] Number of VoIP Modules (0 - 2)
Data[1] Number of channels supported (MSB)
Data[2] Number of channels supported (LSB)
Data[3] Installed VoIP Modules (Bit Field):
 Bit 0 VoIP Module 0 (0 = not installed, 1 = installed)
 Bit 1 VoIP Module 1 (0 = not installed, 1 = installed)
 Bit 2 VoIP Module 2 (0 = not installed, 1 = installed)

Note that for Data 4 - 6 each module can be in one of the following states:
0x02 - Not Present, 0x05 - In Service, 0x06 - Out of Service, and 0x08 - Failed

Data[4] Module 0 State
Data[5] Module 1 State
Data[6] Module 2 State

Note that for Data 7 - 9 each module can be to one of the following IP resource profiles:

0x01 - G.711 Only (16 spans per module)
0x02 - G.711 + G.723.1 + G.729 + T.38 (8 spans per module)
0x0F - Not applicable (For example, module is not present)

Data[7] Module 0 Profile
Data[8] Module 1 Profile
Data[9] Module 2 Profile

Data[10] Port Consumption Mode
 1 = Non Port-Consuming
 2 = Port-Consuming

Data[11-14] Refer to IPN-3 card

Data[15] Reserved (Set to 0)

IP Network Interface Series 2 Card Module

Data[0] Total number of channels supported for the module (MSB)
Data[1] Total number of channels supported for the module (LSB)
Data[2] Module State
Data[3] Module Profile
Data[4] Bitmap of installed DSPs on a VoIP module (1 - Installed)
Data[5] Bitmap of available DSPs on a VoIP module (1 - available)
Data[6] Number of channels per available DSP
Data[7] MII Bridge FPGA Version
Data[8 - 15] Reserved (Set to 0)

IP Network Interface Series 3 Card

Data[0] Number of VoIP Modules (0 - 1)
Data[1] Number of channels supported (MSB)
Data[2] Number of channels supported (LSB)
Data[3] Installed VoIP Modules (Bit Field):

Bit 0 VoIP Module 0 (0 = not installed, 1 = installed)
Bit 1 VoIP Module 1 (0 = not installed, 1 = installed)

Data[4] Module 0 State
Data[5] Module 1 State

Note that for Data 4 - 5 each module can be in one of the following states:

0x02 - Not Present, 0x05 - In Service, 0x06 - Out of Service, and 0x08 - Failed

Data[6] Module 0 Profile
Data[7] Module 1 Profile

Note that for Data 6-7 each module can be to one of the following IP resource profiles:

0x04 - G.711 Only (16 spans per module)
0x05 - LBR + T.38 (8 spans per module)
0x0F - Not applicable (For example, module is not present)

Data[8] Port Consumption Mode
1 = Non Port-Consuming
2 = Port-Consuming

Data[9] Main board dip switch settings (bit mask) (first value is default)
Bit 2-3 Speed
0 0 = 10Mbps
0 1 = 100Mbps
1 0 = 1000Mbps
Bits [4-7] Reserved (should be zero)

Data[10] Port 0 Info (CTRL 0)
Bit 0 Ethernet Link Status
 0 = Ethernet link down
 1 = Ethernet link up
Bit 1 Duplex Type
 0 = Half Duplex
 1 = Full Duplex
Bit 2-3 Speed
 0 0 = 10Mbps
 0 1 = 100Mbps
 1 0 = 1000Mbps

Bits [4-7] Reserved (should be zero)

Data[11] Port 1 Info (CTRL 1)
Bit 0 Ethernet Link Status
 0 = Ethernet link down
 1 = Ethernet link up
Bit 1 Duplex Type
 0 = Half Duplex
 1 = Full Duplex

Data[12] Port 2 Info (DATA 0)
Bit 0 Ethernet Link Status
 0 = Ethernet link down
 1 = Ethernet link up
Bit 1 Duplex Type
 0 = Half Duplex
 1 = Full Duplex
Bit 2-3 Speed
 0 0 = 10Mbps
 0 1 = 100Mbps
 1 0 = 1000Mbps
Bits [4-7] Reserved (should be zero)

Data[13] Port 3 Info (DATA 1)
Bit 0 Ethernet Link Status
 0 = Ethernet link down
 1 = Ethernet link up
Bit 1 Duplex Type
 0 = Half Duplex
 1 = Full Duplex
Bit 2-3 Speed
 0 0 = 10Mbps
 0 1 = 100Mbps
 1 0 = 1000Mbps
Bits [4-7] Reserved (should be zero)

Bit 0 Ethernet Link Status
 0 = Ethernet link down
 1 = Ethernet link up
Bit 1 Duplex Type
 0 = Half Duplex
 1 = Full Duplex

	<p>Bit 2-3 Speed 0 0 = 10Mbps 0 1 = 100Mbps 1 0 = 1000Mbps Bits [4-7] Reserved (should be zero)</p> <p>Data[14] Port 4 Info (DATA 2) Bit 0 Ethernet Link Status 0 = Ethernet link down 1 = Ethernet link up Bit 1 Duplex Type 0 = Half Duplex 1 = Full Duplex Bit 2-3 Speed 0 0 = 10Mbps 0 1 = 100Mbps 1 0 = 1000Mbps Bits [4-7] Reserved (should be zero)</p> <p>Data[15] Port 5 Info (DATA 3) Bit 0 Ethernet Link Status 0 = Ethernet link down 1 = Ethernet link up Bit 1 Duplex Type 0 = Half Duplex 1 = Full Duplex Bit 2-3 Speed 0 0 = 10Mbps 0 1 = 100Mbps 1 0 = 1000Mbps Bits [4-7] Reserved (should be zero)</p> <p><u>IP Network Interface Series 3 Card Module</u></p> <p>Data[0] Total number of channels supported for the module (MSB) Data[1] Total number of channels supported for the module (LSB) Data[2] Module State Data[3] Module Profile Data[4] Bitmap of installed DSPs on a VoIP module (1 - Installed) Data[5] Bitmap of available DSPs on a VoIP module (1 - available) Data[6] Number of channels per available DSP</p>
:	<p>MAC Addresses Data[0,1] Reserved Data[2-5] 1C 03 35 51 MAC Address for Ethernet Port A Data[6-9] 1C 03 35 51 MAC Address for Ethernet Port A Data[10-17] Reserved</p>
:	<p>Card ID This field contains additional information about identification for the card.</p>
:	<p>RAM Build Number This field indicates the application build number of the system software.</p>

:	Configuration Tag MSB, LSB The configuration tag number assigned by the Tag Configuration message.
:	CPU Speed This field applies to cards other than CSP Matrix Series 3 Cards. If the system is reporting on a CSP Matrix Series 3 Card, this field is not included in the report. 0x00 16 Mhz 0x01 16 Mhz 0x02 20 Mhz 0x03 20 Mhz 0x04 25 Mhz 0x05 50 Mhz 0x08 66 Mhz 0x09 83 Mhz 0x0A 100 Mhz
:	Software Reset Reason This field is reserved for Dialogic use only (it applies to cards other than matrix cards. For example, if the system is reporting on a CSP Matrix Series 3 Card, this field is not included in the report).
:	Checksum

Card Status Report 0x00A6

SwitchKit Name	CardStatusReport
Type	EXS API and SwitchKit API message
Description	<p>Card Status Report 0x00A6</p> <p>This message reports information about a card's status. Because this message is defined differently for different card types, some fields listed below may not apply.</p> <p>This message is sent to the host when the host sends the <i>Card Status Query</i> message or when a card is reset.</p>
Sent by	CSP
Resent in EXS API	This message is resent once after five (5) seconds.
Example Message for EXS API	<p>The following example shows a trace of a <i>Card Status Report</i> message for a 4-span ST1LC card.</p> <p>Header (bytes 0-9):</p> <pre>fe [00 51] [00 a6] 00 06 ff [00 00]</pre> <p>AIB (bytes 10-14):</p> <pre>00 01 01 01 01</pre> <p>Misc. Info. (bytes 15-27):</p> <pre>0c 12 0f 42 09 01 04 04 01 [02 16] [08 00]</pre> <p>Card Information (bytes 28-61):</p> <pre>00 08 / 0c 02 ff ff / 0c 02 ff ff / 0c 02 ff ff / 0c 02 ff ff / 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</pre> <p>Hardware Configuration (bytes 62-77):</p> <pre>[00 00] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</pre> <p>Misc. Info. (bytes 78-84):</p> <pre>0c 00 [00 00] 02 0e cs</pre>
SwitchKit Code	<p>C Structure</p> <pre>typedef struct { unsigned short Status; UBYTE AddrInfo[30]; UBYTE CardType; UBYTE CardStatus; UBYTE ConfTestRslts;</pre>

```

    UBYTE ArtRevision;
    UBYTE FunctRevision;
    UBYTE ROMMajorRevision;
    UBYTE ROMMinorRevision;
    UBYTE RAMMajorRevision;
    UBYTE RAMMinorRevision;
    unsigned short SerialNumber;
    unsigned short RAMSize;
    UBYTE CardSpecifics[384];
    UBYTE HardwareConfig[16];
    UBYTE CardID;
    UBYTE RAMBuildNum;
    unsigned short ConfigTag;
    UBYTE LineCardSpeed;
    UBYTE SoftwareResetReason;
} XL_CardStatusReport;

```

C++ Class

```

class XLC_CardStatusReport : public XLC_InboundMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getModule() const;
    void setModule(UBYTE x);
    UBYTE getCardType() const;
    void setCardType(UBYTE x);
    UBYTE getCardStatus() const;
    void setCardStatus(UBYTE x);
    UBYTE getConfTestRslts() const;
    void setConfTestRslts(UBYTE x);
    UBYTE getArtRevision() const;
    void setArtRevision(UBYTE x);
    UBYTE getFunctRevision() const;
    void setFunctRevision(UBYTE x);
    UBYTE getROMMajorRevision() const;
    void setROMMajorRevision(UBYTE x);
    UBYTE getROMMinorRevision() const;
    void setROMMinorRevision(UBYTE x);
    UBYTE getRAMMajorRevision() const;
    void setRAMMajorRevision(UBYTE x);
    UBYTE getRAMMinorRevision() const;
    void setRAMMinorRevision(UBYTE x);
    unsigned short getSerialNumber() const;
    void setSerialNumber(unsigned short x);
    unsigned short getRAMSize() const;
    void setRAMSize(unsigned short x);
    const UBYTE *getCardSpecifics() const;
    UBYTE *getCardSpecifics();
    void setCardSpecifics(UBYTE *x);
    const UBYTE *getHardwareConfig() const;
    UBYTE *getHardwareConfig();
    void setHardwareConfig(UBYTE *x);
    UBYTE getCardID() const;

```

```

void setCardID(UBYTE x);
UBYTE getRAMBuildNum() const;
void setRAMBuildNum(UBYTE x);
unsigned short getConfigTag() const;
void setConfigTag(unsigned short x);
UBYTE getLineCardSpeed() const;
void setLineCardSpeed(UBYTE x);
UBYTE getSoftwareResetReason() const;
void setSoftwareResetReason(UBYTE x)
    
```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0005)
3, 4	Message Type (0x00A6)	3, 4	Message Type (0x00A6)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8, 9	Status MSB, LSB	8	Checksum
:	AIB		
:	Card Type		
:	Card Status		
:	Confidence Test Results		
:	PC Artwork Revision		
:	Functional Revision		
:	ROM Major Revision		
:	ROM Minor Revision		
:	RAM Major Revision		
:	RAM Minor Revision		
:	Serial Number MSB, LSB		
:	RAM Size MSB, LSB		
:	Card Information (Field length varies)		
:	Hardware Configuration		
:	Card ID		
:	RAM Build Number		
:	Configuration Tag MSB, LSB		
:	CPU Speed		
:	Software Reset Reason		
:	Checksum		

**EXS API Hex Format-
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0005)
3, 4	Message Type (0x00A6)	3, 4	Message Type (0x00A6)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8, 9	Status MSB, LSB	8	Checksum
:	AIB Address Method 0x00 - Individual AEs <hr/> Number of AEs to follow <hr/> AE 0x01 Slot		
:	Card Type 0x00 8-Span ST1LC 0x01 4-Span ST1LC 0x03 MFDSP 0x0A SS7 0x0B SE1LC 4-Span 0x0C SE1LC 8-Span 0x0E SJ1LC 4-Span 0x13 ISDN PRI 0x14 DASS 2 0x15 DSP-ONE 0x16 T-ONE 8-Span 0x17 E-ONE 8-Span 0x18 T-ONE 16-Span 0x19 E-ONE 16-Span 0x1D J-ONE 8-Span 0x1E J-ONE 16-Span 0x1F DS3 0x33 Subrate Controller		

Card Status Report 0x00A6

0x42	EXNET Connect PCI H.100
0x50	SS7 PQ
0x54	EXNET-ONE
0x60	VDAC ONE
0x61	VDAC VoIP Module
0x65	IP Network Interface Series 2
0x6B	DSP Series 2
0x6D	IP Network Interface Series 2 VoIP Module
0x71	SS7 Series 3
0x72	CSP 2000 Matrix I/O
0x73	CSP 2000 Matrix
0x75	CSP 1000 Matrix
0x76	ISDN Series 3
0x79	IP Signaling Series 3 Card
0x80	Virtual VDAC
0x90	DSP Series 2 Plus
0x91	IP Network Interface Series 3
0x92	Multi-Function Media I/O Plus
0xAA	Illegal I/O
0xCA	Multi-Function Media I/O
0xCC	DS3 Standby I/O
0xCF	VDAC I/O
0xD2	Virtual T1 16-span
0xD5	Virtual E1/J1 16-Span
0xD8	CCS I/O Series 3
0xDB	J-ONE Redundant I/O
0xDC	J-ONE Standby I/O
0xDD	DS3 I/O
0xDF	CCS I/O
0xE0	E-ONE 75 Ohm Redundant I/O
0xE1	T-ONE 100 Ohm Redundant I/O
0xE2	E-ONE 120 Ohm Redundant I/O
0xE3	E-ONE 75 Ohm Standby I/O
0xE4	T-ONE 100 Ohm Standby I/O
0xE5	E-ONE 120 Ohm Standby I/O
0xE8	SS7 Multi-Protocol I/O
0xF0	Power Supply Card
0xF5	EXNET Ring I/O
0xF3	Matrix I/O
0xF6	Rear Fan Tray
0xF7	SS7 Redundant I/O
0xF8	Front Fan Tray
0xF9	ISDN Redundant I/O
0xFA	Midplane
0xFE	Dead Card: All subsequent data fields do not apply and all other bytes in the message should be ignored.
0xFF	Empty Slot: All subsequent data fields do not apply.

:	<p>Card Status</p> <p>This field is a bit mask that applies to the following cards:</p> <p>CSP Matrix Series 3 Card, T-ONE, E-ONE, J-ONE, DS3, EXNET-ONE, SS7-PQ, SS7 Series 3, DSP-ONE, MFDSP, DSP Series 2, DSP Series 2 Plus, ISDN PRI, ISDN Series 3, ST1LC, SE1LC, SJ1LC, VDAC-ONE, IP Network Interface Series 2 and 3 *, IP Signaling Series 3.</p> <p>Bit</p> <p>0–1 Reset Reason</p> <ul style="list-style-type: none">0 Dialogic use only1 Power-up2 Dialogic use only3 Reset button pushed <p>2–3 Card State</p> <ul style="list-style-type: none">0 In service1 The CSP has taken the card out of service2 The host has taken the card out of service3 The host has put the card in a standby state <p>4 Invalid battery-backed configuration data. The host must reconfigure the card</p> <p>5 Faults are logged Faults may be obtained using a <i>Fault Log Query</i> message. Obtain a Fault Log and report it to Dialogic Technical Support as soon as possible.</p> <p>6 Failed confidence tests. Information about the confidence test which failed is in the "Confidence Test Results" field.</p> <p>7 General hardware failure The card is off-line and must be serviced. For the ISDN PRI card, see the Hardware Configuration Field for additional information.</p> <p>* If the IP Network Interface Series 2 and 3 card is out of service, ignore the card state for the modules.</p>
---	---

	<p>Fan Tray</p> <p>Bit</p> <p>0 Reserved</p> <p>1 Fan 6 failed</p> <p>2 Fan 5 failed</p> <p>3 Fan 4 failed</p> <p>4 Fan 3 failed</p> <p>5 Fan 2 failed</p> <p>6 Fan 1 failed</p> <p>7 General hardware failure. The card is off-line and must be serviced.</p> <p>Power Supply Card</p> <p>Bit</p> <p>0 Module 3 failed</p> <p>1 Module 3 in tolerance</p> <p>2 Module 2 failed</p> <p>3 Module 2 in tolerance</p> <p>4 Module 1 failed</p> <p>5 Module 1 in tolerance</p> <p>6 Out of Tolerance</p> <p>7 General hardware failure. The card is off-line and must be serviced.</p> <p>DS3, E1 and T1 Redundant and Standby I/O Cards</p> <p>Bit</p> <p>0–5 Reserved</p> <p>6 Output Enabled</p> <p>7 Relays Enabled (0=Normal, 1=Switchover)</p> <p>Matrix I/O Card</p> <p>Bit</p> <p>0 Interface B link state: 0=Down, 1=Up</p> <p>1 Interface A link state: 0=Down, 1=Up</p> <p>2 The active Ethernet Interface: 0=Interface A. 1=Interface B</p>
:	<p>Confidence Test Results</p> <p>This field is bit-defined. It applies to the CSP Matrix Series 3 CardS and line cards.</p> <p>Bit</p> <p>0–5 Reserved</p> <p>6 Bit set indicates RAM test failure</p> <p>7 Bit set indicates ROM checksum failure</p>
:	<p>PC Artwork Revision</p> <p>An ASCII character in the range "A" to "Z" (0x41–0x5B) that indicates the revision level of the card's Printed Circuit Artwork.</p>
:	<p>Functional Revision</p> <p>The card's functional revision level</p>
:	<p>ROM Major Revision</p> <p>The major revision level of the firmware running on the card.</p>
:	<p>ROM Minor Revision</p> <p>The minor revision level of the firmware running on the card.</p>
:	<p>RAM Major Revision</p> <p>The major revision level of the system software running on the card.</p>
:	<p>RAM Minor Revision</p> <p>The minor revision level of the system software running on the card.</p>

:	<p>Serial Number MSB, LSB This two-byte field indicates the serial number of the card.</p>
:	<p>RAM Size MSB, LSB This two-byte field indicates the amount of RAM on the card, in 1K increments. For the VDAC VoIP Card and VDAC VoIP Module, RAM is measured in 1 MB increments.</p>
:	<p>Card Information (Field length varies) The card information, including the length of the field, varies per card type. The system uses the value 0x00 for fields that do not apply to the card or for all fields, if there is no card information.</p> <p>IP Signaling Series 3 Card No card specific or hardware specific information is sent for this card.</p> <p>T-ONE, E-ONE, J-ONE, DS3, ST1LC, SE1LC, VDAC-ONE, IP Network Interface Series 2 and 3, and SS7 Multi-Protocol cards</p> <p>The card information for these cards indicates the logical spans assigned to the card. The logical spans are indicated in the Logical Span AIB (0x0C) as shown below. The CSP uses the value 0xFF to indicate that no span has been assigned. An AIB for a 4-span line card has four address elements and the value 0x00 for the remaining bytes. An AIB for an 8-span line card has eight address elements and the value 0x00 for the remaining bytes.</p> <p>Data[0] Address Method (0x00, Single Entity) Data[1] Number of Address Elements (equal to number of spans)</p> <p>Address Element 1: Span 0 Data[2] Address Type (0x0C, Logical Span) Data[3] Data Length (0x02) Data[4] Logical Span ID, MSB Data[5] Logical Span ID, LSB</p> <p>Address Element N: Span N Data[...] Address Type (0x0C, Logical Span) Data[...] Data Length (0x02) Data[...] Logical Span ID, MSB Data[...] Logical Span ID, LSB</p>

DSP Series 2 and DSP Series 2 Plus Cards

The card information for the DSP Series 2 and DSP Series 2 Plus cards indicates the function types for all DSPs on all Modules on the card, because each DSP can do multiple configurations.

<u>Byte</u>	<u>Function Type</u>
Data[0]	Module 0 / DSP 0 / RCV 0
Data[1]	Module 0 / DSP 0 / TXMT 0
Data[2]	Module 0 / DSP 0 / RCV 1
Data[3]	Module 0 / DSP 0 / TXMT 1
Data[4]	Module 0 / DSP 1 / RCV 0
Data[5]	Module 0 / DSP 1 / TXMT 0
Data[6]	Module 0 / DSP 1 / RCV 1
Data[7]	Module 0 / DSP 1 / TXMT 1
Data[8]	Module 0 / DSP 2 / RCV 0
Data[9]	Module 0 / DSP 2 / TXMT 0
Data[10]	Module 0 / DSP 2 / RCV 1
Data[11]	Module 0 / DSP 2 / TXMT 1
Data[12]	Module 0 / DSP 3 / RCV 0
Data[13]	Module 0 / DSP 3 / TXMT 0
Data[14]	Module 0 / DSP 3 / RCV 1
Data[15]	Module 0 / DSP 3 / TXMT 1
Data[16]	Module 1 / DSP 0 / RCV 0
Data[17]	Module 1 / DSP 0 / TXMT 0
Data[18]	Module 1 / DSP 0 / RCV 1
Data[19]	Module 1 / DSP 0 / TXMT 1
Data[20]	Module 1 / DSP 1 / RCV 0
Data[21]	Module 1 / DSP 1 / TXMT 0
Data[22]	Module 1 / DSP 1 / RCV 1
Data[23]	Module 1 / DSP 1 / TXMT 1
Data[24]	Module 1 / DSP 2 / RCV 0
Data[25]	Module 1 / DSP 2 / TXMT 0
Data[26]	Module 1 / DSP 2 / RCV 1
Data[27]	Module 1 / DSP 2 / TXMT 1
Data[28]	Module 1 / DSP 3 / RCV 0
Data[29]	Module 1 / DSP 3 / TXMT 0
Data[30]	Module 1 / DSP 3 / RCV 1
Data[31]	Module 1 / DSP 3 / TXMT 1
Data[32]	Reserved
Data[33]	Reserved

EXNET-ONE

This card generates no special information for this field. All bytes are 0x00.

SS7 Card

Data[0] Maximum Configurable Links for card
Data[1] Slot Number of Active SS7 Card (see Note)
Data[2] Slot Number of Secondary SS7 Card in Redundant Pair (see Note)
Data[3] Stack ID
Data[4] Stack ID
Data[5] Stack ID
Data[6] Stack ID
Data[7–15]Reserved

NOTE: Do not try to determine the redundancy status from bytes Data[1] and Data[2].
To determine the redundancy status, use the *CCS Redundancy Query* message.

Redundant I/O and Standby I/O

A Slot AIB indicates the slot number of the other I/O (Standby I/O or Redundant I/O)
involved in a switchover.

SS7 Multi-Protocol I/O Card

0x0C Logical Span AIB

PCI H.100 Card

The card information fields depend upon on H.100 bus clock speed and the number of spans per
channel.

4MHz and 24 channels per span display 21 span entries
4MHz and 32 channels per span display 16 span entries
8MHz and 24 channels per span display 80 span entries
8MHz and 32 channels per span display 64 span entries

:	<p>Hardware Configuration (16 bytes) This 16-byte field describes the hardware configuration of the card specified. This field applies to only the cards described below. It also describes DIP switch positions on the cards indicated.</p> <p><u>DS3 cards</u></p> <p>Data[0] Number of Spans Supported: 0x18 24 Spans 0x1C 28 Spans</p> <p>Data[1] DS3 Offset Framer Status Bit 0 - DS3 Offset 0 Framer Status 0 - Out of Service 1 - In Service (default) Bit [1-7] Reserved</p> <p>Data [2-13] Reserved</p> <p>Data[14] Hardware Switch 1 Settings (also refer to the Hardware Product Descriptions) (the right-most bit is dip 1)</p> <p>Data[15] Total Available Spans</p> <p><u>ST1LC and SE1LC cards</u></p> <p>Data[0] Informs the host of primary and secondary loop timing clock sources configured on the card. Bits 0-2 If secondary loop timing is enabled, this field contains the offset of the span for which it is enabled. Bit 3 Secondary Loop timing 0 Disabled 1 Enabled Bit 4-6 If primary loop timing is enabled, this field contains the offset of the span for which it is enabled. Bit 7 Primary Loop timing 0 Disabled 1 Enabled</p> <p>Data[1-15] Reserved</p>
---	---

T-ONE, E-ONE, J-ONE cards

Data 0–3 informs the host of primary and secondary loop timing clock sources configured on the card

Data[0] Primary Loop Timing
0 Not Enabled
1 Enabled

Data[1] If Primary Loop Timing is enabled, this field contains the span offset of the span for which it is enabled.

Data[2] Secondary Loop Timing
0 Not enabled
1 Enabled

Data[3] If Secondary Loop Timing is enabled, this field contains the span offset of the span for which it is enabled.

Data[4–13] Reserved

Data[14] Hardware Switch 1 Settings (also refer to the Hardware Product Descriptions) (the right-most bit is dip 1)

Data[15] Total Available Spans

DSP Series 2 and DSP Series 2 Plus Cards

This field informs the host of the number and types of DSP Modules installed on the card.

- Data[0] Number of Modules installed on the card
- Data[1] Reserved
- Data[2] Module 0 Type
 - 0x6C DSP 2 Module
 - 0x03 No Module installed
- Data[3] Module 1 Type
 - 0x6C DSP 2 Module
 - 0x03 No Module installed
- Data[4, 5] Reserved
- Data 6-13: DSP Chip States
 - 0x00 In Service
 - 0x01 Out of Service (or No Module Installed)
- Data[6] Module 0 / Chip 0 State
- Data[7] Module 0 / Chip 1 State
- Data[8] Module 0 / Chip 2 State
- Data[9] Module 0 / Chip 3 State
- Data[10] Module 1 / Chip 0 State
- Data[11] Module 1 / Chip 1 State
- Data[12] Module 1 / Chip 2 State
- Data[13] Module 1 / Chip 3 State
- Data[14] Main Board DIP Switch S1Settings (Bit Field)
 - Bit 0 Debug 1 Mode (ON=Disabled; OFF= Enabled)
 - Bit 1 Reserved (ON)
 - Bit 2 Debug 2 Mode (ON=Disabled; OFF= Enabled)
 - Bit 3 Ethernet Link Mode (ON=Auto-Negotiate; OFF= Force 100 Mbps/
Full Duplex)
 - Bit 4 Reserved (ON)
 - Bit 5 Reserved (ON)
 - Bit 6 Reserved (ON)
 - Bit 7 Reserved (ON)
 - Note: Each Bit maps to the physical DIP switch positions (Bit 0 to Position 1
- Bit 7 to Position 8)
 - Note: Default setttings are ON position
- Data[15] Reserved

CSP Matrix Series 3 Card

Data[0] Main Board DIP Switch S1Settings (Bit Field)
Bit 0 Print to Terminal (ON=Disabled; OFF= Enabled)
Bit 1 Reserved (ON)
Bit 2 System Software/Probe (ON=Enables System Software; OFF Enables Probe)
Bit 3 CSP Call Control/Services (ON=Call Control Call Processing Enabled; OFF=Services Call processing Enabled)
Bit 4 Reserved (ON)
Bit 5 Reserved (ON); OFF=API
Bit 6 Using Second Ethernet Port (ON=At startup, Ethernet Port 1 (ETH1) is only used; OFF=At startup, both Ethernet Ports (ETH1 and ETH2) can be used.
Bit 7 No Static IP Address (ON=If RARP and BOOTP can not obtain an IP address, the static IP address is used; OFF=If RARP and BOOTP can not obtain an IP address, the static IP address is **not** used)
Note: Each Bit maps to the physical DIP switch positions (Bit 0 to Position - Bit 7 to Position 8)
Note: Default settings are ON position, except Bit 5 which is OFF

Data[1] Main Board DIP Switch S2Settings (Bit Field)
Bit 0 Ethernet Link Mode (ON= Force 100 Mbps/Full Duplex; OFF=Auto-Negotiate)
Bit 1 Reserved (OFF)
Bit 2 Reserved (OFF)
Bit 3 Reserved (OFF)
Bit 4 Reserved (OFF)
Bit 5 Reserved (OFF)
Bit 6 Reserved (OFF)
Bit 7 Reserved (OFF)
Note: Each Bit maps to the physical DIP switch positions (Bit 0 to Position 1 - Bit 7 to Position 8)
Note: Default settings are OFF position

IP Signaling Series 3 Card

Data[0] Main Board DIP Switch SW1Settings (Bit Field)
Bit 0 Debug 1 Mode (ON=Disabled; OFF= Enabled)
Bit 1 Monitor Port (ON= Monitor Port setting is 19200 baud and is and not selectable; OFF=Reserved)
Bit 2 Debug 2 Mode (ON=Disabled; OFF= Enabled)
Bit 3 Ethernet Link Mode (ON=Auto-Negotiate; OFF= Force 100 Mbps/ Full Duplex)
Bit 4 Layer 5 (ON=Disabled; OFF= Enabled)
Bit 5 Reserved (ON); API
Bit 6 Reserved (ON)
Bit 7 No Static IP Address (ON=If RARP and BOOTP can not obtain an IP address, the static IP address is used; OFF=If RARP and BOOTP can not obtain an IP address, the static IP address is **not** used)
Note: Each Bit maps to the physical DIP switch positions (Bit 0 to Position 1 - Bit 7 to Position 8)
Note: Default settings are ON position, except Bit 5 which is OFF

SS7 Series 3 Card

Data[0] Main Board DIP Switch SW1Settings (Bit Field)
Bit 0 Debug 1 Mode (ON=Disabled; OFF= Enabled)
Bit 1 Monitor Port (ON/OFF= Monitor Port setting is 19200 baud)
Bit 2 Debug 2 Mode (ON=Disabled; OFF= Enabled)
Bit 3 Reserved (ON)
Bit 4 Reserved (ON)
Bit 5 Reserved (ON)
Bit 6 Reserved (ON)
Bit 7 Reserved (ON)
Note: Each Bit maps to the physical DIP switch positions (Bit 0 to Position 1
- Bit 7 to Position 8)
Note: Default setttings are ON position

ISDN Series 3 Card

Data[0] Main Board DIP Switch SW1Settings (Bit Field)
Bit 0 Debug 1 Mode (ON=Disabled; OFF= Enabled)
Bit 1 Monitor Port (ON/OFF= Monitor Port setting is 19200 baud)
Bit 2 Debug 2 Mode (ON=Disabled; OFF= Enabled)
Bit 3 Reserved (ON)
Bit 4 Reserved (ON)
Bit 5 Reserved (ON)
Bit 6 Reserved (ON)
Bit 7 Reserved (ON)
Note: Each Bit maps to the physical DIP switch positions (Bit 0 to Position 1
- Bit 7 to Position 8)
Note: Default setttings are ON position

EXNET-ONE card

Data[0-3] Informs host of primary and secondary loop timing clock sources configured from ring timing on the card. For EXNET-ONE cards configured as a Single Ring Redundancy Standby, the Loop Timing fields are always disabled.

Data[0] Primary Loop Timing
0x00 Not Enabled
0x01 Enabled
Data[1] Reserved
Data[2] Secondary Loop Timing
0x00 Not Enabled
0x01 Enabled
Data[3] Reserved

SS7 cards

The system does not report hardware configuration information for these cards.
All bytes are 0x00.

SS7 Multi-Protocol I/O card

Data[0] Channel 0 Data Rate
0x00 64 Kbps
0x01 56 Kbps, for non-V.35 applications
0x80 V.35, 56 Kbps
0x81 48 Kbps, for non-V.35 applications
0xC0 V.35, 48 Kbps

Data[1] Channel 0 Electrical Interface
0x00 V.35, DTE
0x01 V.35, DCE

Data[2] Channel 1 Data Rate (same possibilities as Channel 0, above)
Data[3] Channel 1 Electrical Interface (same possibilities as Channel 0, above)
Data[4] Channel 2 Data Rate (same possibilities as Channel 0, above)
Data[5] Channel 2 Electrical Interface (same possibilities as Channel 0, above)
Data[6] Channel 3 Data Rate (same possibilities as Channel 0, above)
Data[7] Channel 3 Electrical Interface (same possibilities as Channel 0, above)

VDAC-ONE card

Data[0] Number of VoIP Modules (1 - 4)
Data[1] Number of channels supported (MSB)
Data[2] Number of channels supported (LSB)
Data[3] Installed VoIP Modules (Bit Field):
 Bit 0 VoIP Module 0 (0 = not installed, 1 = installed)
 Bit 1 VoIP Module 1 (0 = not installed, 1 = installed)
 Bit 2 VoIP Module 2 (0 = not installed, 1 = installed)
 Bit 3 VoIP Module 3 (0 = not installed, 1 = installed)
Data[4] Module 0 state (0x05=ready, other values=not ready)
Data[5] Module 1 state (0x05=ready, other values=not ready)
Data[6] Module 2 state (0x05=ready, other values=not ready)
Data[7] Module 2 state (0x05=ready, other values=not ready)
Data[8] Port Consumption Mode
 1 Reserved
 2 Port-Consuming
Data[9-15] Reserved (should be zero)

	<p><u>VDAC VoIP Module</u></p> <p>Data[0] Total number of channels supported on the module Data[1] Number of channels supported for DSP 0 Data[2] Number of channels supported for DSP 1 Data[3] Number of channels supported for DSP 2 Data[4] Number of channels supported for DSP 3 Data[5] Number of channels supported for DSP 4 Data[6] Number of channels supported for DSP 5 Data[7] Number of channels supported for DSP 6 Data[8] Number of channels supported for DSP 7 Data[9-15]Reserved (should be zero)</p> <p><u>IP Network Interface Series 2 card</u></p> <p>Data[0] Number of VoIP Modules (0 - 2) Data[1] Number of channels supported (MSB) Data[2] Number of channels supported (LSB) Data[3] Installed VoIP Modules (Bit Field): Bit 0 VoIP Module 0 (0 = not installed, 1 = installed) Bit 1 VoIP Module 1 (0 = not installed, 1 = installed) Bit 2 VoIP Module 2 (0 = not installed, 1 = installed)</p> <p>Note that for Data 4 - 6 each module can be in one of the following states: 0x02 - Not Present, 0x05 - In Service, 0x06 - Out of Service, and 0x08 - Failed</p> <p>Data[4] Module 0 State Data[5] Module 1 State Data[6] Module 2 State</p> <p>Note that for Data 7 - 9, each module can be for one of the following IP resource profiles: 0x01 - G.711 Only (16 spans per module) 0x02 - G.711 + G.723.1 + G.729 + T.38 (8 spans per module) 0x0F - Not applicable (For example, module is not present)</p> <p>Data[7] Module 0 Profile Data[8] Module 1 Profile Data[9] Module 2 Profile</p> <p>Data[10] Port Consumption Mode 1 = Non Port-Consuming 2 = Port-Consuming Data [11-14] Refer to IPN-3 card</p> <p>Data[15] Reserved (Set to 0)</p>
	<p><u>IP Network Interface Series 2 card Module</u></p> <p>Data[0] Total number of channels supported for the module (MSB) Data[1] Total number of channels supported for the module (LSB) Data[2] Module State Data[3] Module Profile Data[4] Bitmap of installed DSPs on a VoIP module (1 - Installed) Data[5] Bitmap of available DSPs on a VoIP module (1 - available) Data[6] Number of channels per available DSP Data[7] MII Bridge FPGA Version Data[8 - 15] Reserved (Set to 0)</p>

IP Network Interface Series 3 Card

Data[0] Number of VoIP Modules (0 - 1)

Data[1] Number of channels supported (MSB)

Data[2] Number of channels supported (LSB)

Data[3] Installed VoIP Modules (Bit Field):

Bit 0 VoIP Module 0 (0 = not installed, 1 = installed)

Bit 1 VoIP Module 1 (0 = not installed, 1 = installed)

Data[4] Module 0 State

Data[5] Module 1 State

Note that for Data 4 - 5 each module can be in one of the following states:

0x02 - Not Present, 0x05 - In Service, 0x06 - Out of Service, and 0x08 - Failed

Data[6] Module 0 Profile

Data[7] Module 1 Profile

Note that for Data 6-7 each module can be to one of the following IP resource profiles:

0x04 - G.711 Only (16 spans per module)

0x05 - LBR + T.38 (8 spans per module)

0x0F - Not applicable (For example, module is not present)

Data[8] Port Consumption Mode

1 = Non Port-Consuming

2 = Port-Consuming

Data[9] Main board dip switch settings (bit mask) (first value is default)

Data[10] Port 0 Info (CTRL 0)

Bit 0 Ethernet Link Status

0 = Ethernet link down

1 = Ethernet link up

Bit 1 Duplex Type

0 = Half Duplex

1 = Full Duplex

Bit 2-3 Speed

0 0 = 10Mbps

0 1 = 100Mbps

1 0 = 1000Mbps

Bits [4-7] Reserved (should be zero)

000-1 0000
Data[11] Port 1 Info (CTRL 1)

Bit 0 Ethernet Link Status

0 = Ethernet link down

1 = Ethernet link up

Bit 1 Duplex Type

0 = Half Duplex

1 = Full Duplex

Bit 2-3 Speed

0 0 = 10Mbps

0 1 = 100Mbps

1 0 = 1000Mbps

Bits [4-7] Reserved (should be zero)

	<p>Data[12] Port 2 Info (DATA 0)</p> <ul style="list-style-type: none">Bit 0 Ethernet Link Status<ul style="list-style-type: none">0 = Ethernet link down1 = Ethernet link upBit 1 Duplex Type<ul style="list-style-type: none">0 = Half Duplex1 = Full DuplexBit 2-3 Speed<ul style="list-style-type: none">0 0 = 10Mbps0 1 = 100Mbps1 0 = 1000MbpsBits [4-7] Reserved (should be zero) <p>Data[13] Port 3 Info (DATA 1)</p> <ul style="list-style-type: none">Bit 0 Ethernet Link Status<ul style="list-style-type: none">0 = Ethernet link down1 = Ethernet link upBit 1 Duplex Type<ul style="list-style-type: none">0 = Half Duplex1 = Full DuplexBit 2-3 Speed<ul style="list-style-type: none">0 0 = 10Mbps0 1 = 100Mbps1 0 = 1000MbpsBits [4-7] Reserved (should be zero) <p>Data[14] Port 4 Info (DATA 2)</p> <ul style="list-style-type: none">Bit 0 Ethernet Link Status<ul style="list-style-type: none">0 = Ethernet link down1 = Ethernet link upBit 1 Duplex Type<ul style="list-style-type: none">0 = Half Duplex1 = Full DuplexBit 2-3 Speed<ul style="list-style-type: none">0 0 = 10Mbps0 1 = 100Mbps1 0 = 1000MbpsBits [4-7] Reserved (should be zero) <p>Data[15] Port 5 Info (DATA 3)</p> <ul style="list-style-type: none">Bit 0 Ethernet Link Status<ul style="list-style-type: none">0 = Ethernet link down1 = Ethernet link upBit 1 Duplex Type<ul style="list-style-type: none">0 = Half Duplex1 = Full DuplexBit 2-3 Speed<ul style="list-style-type: none">0 0 = 10Mbps0 1 = 100Mbps1 0 = 1000MbpsBits [4-7] Reserved (should be zero)
--	---

	<p><u>IP Network Interface Series 3 Card Module</u></p> <p>Data[0] Total number of channels supported for the module (MSB)</p> <p>Data[1] Total number of channels supported for the module (LSB)</p> <p>Data[2] Module State</p> <p>Data[3] Module Profile</p> <p>Data[4] Bitmap of installed DSPs on a VoIP module (1 - Installed)</p> <p>Data[5] Bitmap of available DSPs on a VoIP module (1 - available)</p> <p>Data[6] Number of channels per available DSP</p>
:	<p>MAC Addresses</p> <p>Data[0,1] Reserved</p> <p>Data[2-5] 1C 03 35 51 MAC Address for Ethernet Port A</p> <p>Data[6-9] 1C 03 35 51 MAC Address for Ethernet Port A</p> <p>Data[10-17] Reserved</p>
:	<p>Card ID</p> <p>This field contains additional information about identification for the card.</p>
:	<p>RAM Build Number</p> <p>This field indicates the application build number of the system software.</p>
:	<p>Configuration Tag MSB, LSB</p> <p>The configuration tag number assigned by the Tag Configuration message.</p>
:	<p>CPU Speed</p> <p>This field applies to cards other than CSP Matrix Series 3 Cards. If the system is reporting on a CSP Matrix Series 3 Card, this field is not included in the report.</p> <p>0x00 16 Mhz</p> <p>0x01 16 Mhz</p> <p>0x02 20 Mhz</p> <p>0x03 20 Mhz</p> <p>0x04 25 Mhz</p> <p>0x05 50 Mhz</p> <p>0x08 66 Mhz</p> <p>0x09 83 Mhz</p> <p>0x0A 100 Mhz</p>
:	<p>Software Reset Reason</p> <p>This field is reserved for Dialogic use only (it applies to cards other than matrix cards. For example, if the system is reporting on a CSP Matrix Series 3 Card, this field is not included in the report).</p>
:	<p>Checksum</p>

CCS Redundancy Configure 0x005B

SwitchKit Name CCSRedundancyConfig

Type EXS API and SwitchKit API message

Description **CCS Redundancy Configure 0x005B**

This message configures a chassis for common channel signaling (CCS) redundancy. When the system is successfully configured, all configuration and call processing is mirrored on the secondary card. If the primary card fails, the secondary card resumes call processing where the primary left off. You enter the slot number of the primary and secondary SS7 or ISDN cards (using an AIB), and a configuration option to suspend synchronization or to not suspend synchronization.

Related Message *CCS Redundancy Report 0x0073* (CCSRedundancyReport)



CAUTION

Dialogic strongly recommends sending this message during minimal call processing loads only. Replicating database and call processing information entails heavy processor bandwidth, which affects call processing performance and throughput.

Sent by Host

SwitchKit Code **Configuration**

```
CCSRedundancyConfig (
    Node = integer,
    PrimaryCard = integer,
    SecondaryCard = integer,
    RedundancyType = integer);
```

C Structure

```
typedef struct {
    UBYTE PrimaryCard;
    UBYTE SecondaryCard;
    UBYTE RedundancyType;
} XL_CCSRedundancyConfig;
```

C++ Class

```
class XLC_CCSRedundancyConfig : public
    XLC_OutboundMessage {
public:
    UBYTE getPrimaryCard() const;
    void setPrimaryCard(UBYTE x);
```

```

UBYTE getSecondaryCard() const;
void setSecondaryCard(UBYTE x);
UBYTE getRedundancyType() const;
void setRedundancyType(UBYTE x);
};
    
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x005B)	3, 4	Message Type (0x005B)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status (MSB, LSB)
		10	Checksum
:	AIB Address Method 0x00 - Individual AEs Number of AEs to follow AEs 0x01 Slot (Primary Card) Enter the physical slot number of a non-redundant card or the primary card in a redundant pair. 0x01 Slot (Secondary Card) Enter the physical slot number of the secondary card in a redundant pair. If you define this value as 0xFF, the system cancels redundancy and resets the system to recognize the primary card as if it were a single card configuration. For a non-redundant card, this field <i>must</i> be 0xFF.		
:	Options 0x00 Do Not Suspend Synchronization (Default) 0x01 Suspend Synchronization		
:	Checksum		

CCS Redundancy Query 0x006C

SwitchKit Name CCSRedundancyQuery

Type EXS API and SwitchKit API message

Description **CCS Redundancy Query 0x006C**

Use this message to query the redundancy status (primary or secondary) of common channel signaling (CCS) cards installed in the system. If a redundant pair of cards is installed, the status of both cards is reported.

- Related Messages**
- *CCS Redundancy Configure* 0x0073 (CCSRedundancyConfig)
 - *CCS Redundancy Report* 0x005B (CCSRedundancyReport)

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE Slot;
} XL_CCSRedundancyQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE PrimaryCard;
    UBYTE SecondaryCard;
    UBYTE ActiveCard;
    UBYTE RedundancyStatus;
    UBYTE RedundancyType;
} XL_CCSRedundancyQueryAck;
```

C++ Class

```
class XL_CCSRedundancyQuery : public XLC_OutboundMessage
{
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
};
```

C++ Class Response

```
class XL_CCSRedundancyQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getPrimaryCard() const;
    void setPrimaryCard(UBYTE x);
    UBYTE getSecondaryCard() const;
    void setSecondaryCard(UBYTE x);
```

```
UBYTE getActiveCard() const;  
void setActiveCard(UBYTE x);  
UBYTE getRedundancyStatus() const;  
void setRedundancyStatus(UBYTE x);  
UBYTE getRedundancyType() const;  
void setRedundancyType(UBYTE x);  
UBYTE getReserved() const;  
void setReserved(UBYTE x);  
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)																															
Byte	Field Description	Byte	Field Description																														
0	Frame (0xFE)	0	Frame (0xFE)																														
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)																														
3, 4	Message Type (0x006C)	3, 4	Message Type (0x006C)																														
5	Reserved (0x00)	5	Reserved (0x00)																														
6	Sequence Number	6	Same Sequence Number																														
7	Logical Node ID	7	Logical Node ID																														
:	<u>AIB</u> Address Method 0x00 - Individual AEs	8, 9	Status MSB, LSB																														
	Number of AEs to follow																																
	AE 0x01 Slot																																
:	Checksum																																
Response continued below.																																	
:	<u>AIB</u> Address Method 0x00 - Individual AEs	<table border="0"> <thead> <tr> <th><u>Byte</u></th> <th><u>AE Field</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Address Type</td> <td>(0x01)</td> </tr> <tr> <td>1</td> <td>Data Length</td> <td>(0x01)</td> </tr> <tr> <td>2</td> <td>Data[0]</td> <td>Slot of Primary Card</td> </tr> <tr> <td>0</td> <td>Address Type</td> <td>(0x01)</td> </tr> <tr> <td>1</td> <td>Data Length</td> <td>(0x01)</td> </tr> <tr> <td>2</td> <td>Data[0]</td> <td>Slot of Secondary Card</td> </tr> <tr> <td>0</td> <td>Address Type</td> <td>(0x01)</td> </tr> <tr> <td>1</td> <td>Data Length</td> <td>(0x01)</td> </tr> <tr> <td>2</td> <td>Data[0]</td> <td>Slot of Active Card</td> </tr> </tbody> </table>		<u>Byte</u>	<u>AE Field</u>	<u>Description</u>	0	Address Type	(0x01)	1	Data Length	(0x01)	2	Data[0]	Slot of Primary Card	0	Address Type	(0x01)	1	Data Length	(0x01)	2	Data[0]	Slot of Secondary Card	0	Address Type	(0x01)	1	Data Length	(0x01)	2	Data[0]	Slot of Active Card
<u>Byte</u>	<u>AE Field</u>			<u>Description</u>																													
0	Address Type			(0x01)																													
1	Data Length			(0x01)																													
2	Data[0]			Slot of Primary Card																													
0	Address Type			(0x01)																													
1	Data Length			(0x01)																													
2	Data[0]			Slot of Secondary Card																													
0	Address Type			(0x01)																													
1	Data Length			(0x01)																													
2	Data[0]			Slot of Active Card																													
	Number of AEs to follow																																
	0																																
	1																																
	2																																
	0																																
	1																																
	2																																
	0																																
	1																																
	2																																

CCS Redundancy Query 0x006C

:	<p>Redundancy Status</p> <p>0x01 Synchronizing (Primary Receive From Matrix (RFM))</p> <p>0x02 Synchronizing (Primary Send To Matrix (STM))</p> <p>0x03 Both Cards Active (Primary RFM)</p> <p>0x04 Both Cards Active (Primary STM)</p> <p>0x05 Redundancy is being disabled</p> <p>0x14 Primary Card Reset</p> <p>0x15 Secondary Card Reset</p> <p>0x16 Double Reset</p> <p>0x17 Wait for Secondary Query</p> <p>0x18 Wait for Primary Query</p> <p>0x19 Wait for Primary Recovery</p> <p>0x1A Wait for Primary Alive</p> <p>0x1B Secondary Card Reset</p> <p>0x1C Double Reset</p> <p>0x1D Wait for Secondary Query</p> <p>0x1E Wait for Primary Query</p> <p>0x20 Primary card is out-of-service</p> <p>0x21 Wait for Secondary Recovery</p> <p>0x22 Wait for Secondary Alive</p> <p>0x23 Timer expired while waiting for synchronization</p> <p>0x24 Redundancy disabled by the host</p> <p>0x31 Dual internal redundancy failure. (Redundancy disabled. Card synchronizing while links & CICs being deconfigured)</p>
:	Reserved (0x00)
:	Reserved (0x00)
:	Checksum

CCS Redundancy Report 0x0073

SwitchKit Name CCSRedundancyReport

Type EXS API and SwitchKit API message

Description **CCS Redundancy Report 0x0073**

This message informs the host that either a pair of common channel signaling (CCS) cards is available or a switchover is occurring in the system. This is initiated by the CSP redundancy manager on a redundancy state change.

Related Messages

- *CCS Redundancy Configure* 0x005B (CCSRedundancyConfig)
- *CCS Redundancy Query* 0x006C (CCSRedundancyQuery)

Sent by CSP

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE PrimaryCard;
    UBYTE SecondaryCard;
    UBYTE ActiveCard;
    UBYTE RedundancyStatus;
    UBYTE RedundancyType;
} XL_CCSRedundancyReport;
```

C++ Class

```
class XLC_CCSRedundancyReport : public XLC_InboundMessage
{
public:
    UBYTE getPrimaryCard() const;
    void setPrimaryCard(UBYTE x);
    UBYTE getSecondaryCard() const;
    void setSecondaryCard(UBYTE x);
    UBYTE getActiveCard() const;
    void setActiveCard(UBYTE x);
    UBYTE getRedundancyStatus() const;
    void setRedundancyStatus(UBYTE x);
    UBYTE getRedundancyType() const;
    void setRedundancyType(UBYTE x);
    UBYTE getReserved() const;
    void setReserved(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0005)
3, 4	Message Type (0x0073)	3, 4	Message Type (0x0073)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u> Address Method 0x00 - Individual AEs <hr/> Number of AEs 0x03 <hr/> AEs 0x01 Slot (Primary Card) The physical slot number of either the primary or secondary SS7 or ISDN card. The value 0xFF indicates that a primary card is not in-service. 0x01 Slot (Secondary Card) The physical slot number of the secondary SS7 or ISDN card. The value 0xFF indicates that a secondary card is not in service. 0x01 Slot (Active Card) Slot number of the currently active SS7 or ISDN PRI card.	8	Checksum

:	<p>Redundancy Status</p> <p>0x01 Synchronizing (Primary Receive From Matrix (RFM))</p> <p>0x02 Synchronizing (Primary Send To Matrix (STM))</p> <p>0x03 Both Cards Active (Primary RFM)</p> <p>0x04 Both Cards Active (Primary STM)</p> <p>0x19 Double reset, primary recovery in progress</p> <p>0x1A Primary recovery failed, redundancy deconfigured</p> <p>0x1B Double reset, secondary recovery in progress</p> <p>0x1C Secondary recovery failed, redundancy deconfigured</p> <p>0x20* Primary card is out-of-service</p> <p>0x21* Secondary card is out-of-service</p> <p>0x22* Redundant I/O card is removed</p> <p>0x23* Timer expired while waiting for synchronization</p> <p>0x24* Redundancy disabled by the host</p> <p>0x25* Primary card reset during synchronization</p> <p>0x26* Secondary card reset during synchronization</p> <p>0x27* Primary card detected link failure during synchronization state (1 or 2)</p> <p>0x28* Secondary card detected link failure during synchronization state (1 or 2)</p> <p>0x29 Primary card detected link failure to mate while in stable state (3 or 4)</p> <p>0x30 Secondary card detected link failure to mate while in stable state (3 or 4)</p> <p>0x31* Dual internal redundancy failure (card synchronizing while links and CICs are being deconfigured)</p> <p>0x40 Primary CCS I/O card is out-of-service</p> <p>0x41 Secondary CCS I/O is out-of-service</p> <p>* Redundancy is disabled</p>
:	Reserved (0x00)
:	<p>0x01 Slot (Primary Card)</p> <p>The physical slot number of either the primary or secondary SS7 or ISDN card.</p>
:	Checksum

Channel Connection Status Query 0x0001

SwitchKit Name	ChannelConnectionStatusQuery
Type	EXS API and SwitchKit API message
Description	Channel Connection Status Query 0x0001 Use this message to retrieve a channel's connection status.
Sent by	Host
SwitchKit Code	C Structure

```
typedef struct {
    unsigned short Span;
    UBYTE Channel;
} XL_ChannelConnectionStatusQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    unsigned short SpanA;
    UBYTE ChannelA;
    unsigned short SpanB;
    UBYTE ChannelB;
    UBYTE ChannelAState;
} XL_ChannelConnectionStatusQueryAck;
```

C++ Class

```
class XLC_ChannelConnectionStatusQuery : public
    XLC_OneChannelOutbound {
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
};
```

C++ Class Response

```
class XLC_ChannelConnectionStatusQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    unsigned short getSpanA() const;
    void setSpanA(unsigned short x);
    UBYTE getChannelA() const;
    void setChannelA(UBYTE x);
    unsigned short getSpanB() const;
    void setSpanB(unsigned short x);
    UBYTE getChannelB() const;
    void setChannelB(UBYTE x);
    UBYTE getChannelAState() const;
    void setChannelAState(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (MSB, LSB) 0x00NN	1, 2	Length (MSB, LSB) 0x00NN
3, 4	Message Type (0x0001)	3, 4	Message Type (0x0001)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs Number of AEs to follow AEs 0x0D Channel A 0x0D Channel B	8, 9	Status (MSB, LSB) 0x0010 Logical Span ID A, Channel A is connected to Logical Span ID B, Channel B. 0x0015 Logical Span ID A, Channel A is not connected.
:	Checksum	:	AIB Same as message
Response continued below.			
:	Channel A State This field indicates the call processing state for Layer 4: If the preceding Status field is a positive acknowledgement, this field does not apply. 0x00 Out-of-service 0x03 Idle 0x04 Host Connect Wait 0x05 Outsize ACK Network Wait 0x07 Answered/connected 0x08 Layer 4 Release Wait 0x09 Network Release Wait 0x0A Busied Out 0x0B Externally Outsize 0x0C Hold Acknowledge Wait 0x0D Layer 3 Answer Wait 0x0E Layer 4 Answer Wait 0x10 Layer 4 Recall Wait 0x11 Purge Response Wait 0x12 Purge Wait 0x13 Park 0x14 Layer 4 Connection Clear Wait		
:	Checksum		

Channel Parameter Query 0x0080

SwitchKit Name ChannelParameterQuery

Type EXS API and SwitchKit API message

Description **Channel Parameter Query 0x0080**

This message is used to query signaling timers and filters and configuration for Channel Associated Signaling (CAS) channels. The message response is formatted differently depending on the trunk type of the channel.

- PPL Format: E-ONE, T-ONE, SE1LC cards
- Non-PPL: T-ONE or ST1LC card using non-PPL format.

This message handles all DS3 functionality, including enabling and disabling loop back configuration and loop back queries. You must specify the slot number and the DS3 offset. You must take a DS3 span out of service before you can place it into remote loop back.

Do not use this message for Common Channel Signaling (CCS) channels. For ISDN, use the *B Channel Query* message. For SS7, use the *SS7 CIC Query* message.

NOTE: The message format to query PPL channels includes a Response Format field. For backward compatibility, if no value is present, the response will be in non-PPL format.

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short Span;
    UBYTE Channel;
} XL_ChannelParameterQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE TrunkType;
    UBYTE Data[235];
} XL_ChannelParameterQueryAck;
```

C++ Class

```
class XLC_ChannelParameterQuery : public
    XLC_OneChannelOutbound {
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
```

```
};
```

C++ Class Response

```
class XLC_ChannelParameterQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getTrunkType() const;
    void setTrunkType(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x005B)
3, 4	Message Type (0x0080)	3, 4	Message Type (0x0080)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB	8, 9	Status MSB, LSB
:	Response Format	10	Trunk Type
:	Checksum	11	Host Service State
Response continued below.			
12	Reserved		
13	Transmit Idle Signaling		
14	Transmit Out of Service Signaling		
15	Busy Out Enable		
16	Distant End Release Mode		
17	Answer Supervision Mode		
18–37	PPL Protocol Name		
38, 39	R2 Timer 1		
40, 41	R2 Timer 2		
42, 43	R2 Timer 3		
44, 45	R2 Timer 4		
46, 47	DTMF Timer 1		
48, 49	DTMF Timer 2		
50, 51	DTMF Timer 3		
52, 53	DTMF Timer 4		
54, 55	DTMF Timer 5		
56, 57	DTMF Timer 6		

Channel Parameter Query 0x0080

58, 59	MFR1 Timer 1
60, 61	MFR1 Timer 2
62, 63	MFR1 Timer 3
64, 65	MFR1 Timer 4
66, 67	MFR1 Timer 5
68, 69	MFR1 Timer 6
70, 71	MFR1 Timer 7
72, 73	MFR1 Timer 8
74, 75	DP Timer 1
76, 77	DP Timer 2
78, 79	DP Timer 3
80, 81	DP Timer 4
82, 83	DP Timer 5
84, 85	DP Timer 6
86, 87	DP Timer 7
88, 89	DP Timer 8
90, 91	DP Timer 9
92	Local End Release Mode
93	PCM Encoding Format
94	Checksum

EXS API Hex Format- Detailed **PPL Format**

The table below lists the message and response formats to query channels on an E-ONE, SE1LC, or T-ONE card with PPL format.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1	Length, MSB	1	Length, MSB (0x00)
2	Length, LSB (N)	2	Length, LSB (0x5B)
3	Message Type, MSB (0x00)	3	Message Type, MSB (0x00)
4	Message Type, LSB (0x80)	4	Message Type, LSB (0x80)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID

Channel Parameter Query 0x0080

:	<u>AIB</u> Address Method 0x00 - Individual AEs	8, 9	Status MSB, LSB
	Number of AEs to follow		
	AEs 0x0D Channel		
:	Response Format The desired format of the response data from the CSP. NOTE: If no value is entered in this field, the response will be in non-PPL format. 0x00 PPL (T-ONE, E-ONE, SE1LC cards)	10	Trunk Type 0x09 PPL
:	Checksum	11	Host Service State 0x00 Out of Service 0x01 In Service
	Response continued below.		
12	Reserved		
13	Transmit Idle Signaling		
14	Transmit Out of Service Signaling		
15	Busy Out Enable 0x00 Busy Out disabled on this trunk 0x01 Busy Out enabled on this trunk		
16	Distant End Release Mode 0x01 Park 0x02 Release		
17	Answer Supervision Mode 0x00 Propagate Answer To The Distant End 0x01 Notify Host Of Answer 0x02 Propagate Answer To The Distant End And Notify Host Of Answer 0x03 No Answer Supervision (no propagation of answer or notification)		
18-37	PPL Protocol Name		
38, 39	R2 Timer 1		
40, 41	R2 Timer 2		
42, 43	R2 Timer 3		
44, 45	R2 Timer 4		
46, 47	DTMF Timer 1		
48, 49	DTMF Timer 2		
50, 51	DTMF Timer 3		
52, 53	DTMF Timer 4		
54, 55	DTMF Timer 5		
56, 57	DTMF Timer 6		
58, 59	MFR1 Timer 1		
60, 61	MFR1 Timer 2		
62, 63	MFR1 Timer 3		

Channel Parameter Query 0x0080

64, 65	MFR1 Timer 4
66, 67	MFR1 Timer 5
68, 69	MFR1 Timer 6
70, 71	MFR1 Timer 7
72, 73	MFR1 Timer 8
74, 75	DP Timer 1
76, 77	DP Timer 2
78, 79	DP Timer 3
80, 81	DP Timer 4
82, 83	DP Timer 5
84, 85	DP Timer 6
86, 87	DP Timer 7
88, 89	DP Timer 8
90, 91	DP Timer 9
92	Local End Release Mode 0x01 Park 0x02 Release
93	PCM Encoding Format 0x01 μ -law Encoding 0x02 A-law Encoding
94	Checksum

EXS API Hex Format - Non-PPL Format
Detailed

The following is the message and response format for non-PPL channels (ST1LC card or T-ONE card using non-PPL channels).

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1	Length, MSB	1	Length, MSB (0x00)
2	Length, LSB (N)	2	Length, LSB (0xF4)
3	Message Type, MSB (0x00)	3	Message Type, MSB (0x00)
4	Message Type, LSB (0x80)	4	Message Type, LSB (0x80)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB 0x0D Channel	8, 9	Status MSB, LSB
:	Checksum		
Response continued below.			

Channel Parameter Query 0x0080

10	Trunk Type 0x01 E&M 0x02 FXO Loop Start 0x03 FXS Loop Start 0x04 FXO Ground Start 0x05 FXS Ground Start 0x06 DPO 0x07 DPT 0x08 Reserved 0x0A DID
11	Flash Timing Status 0x00 Flash Timing OFF 0x01 Flash Timing ON, propagate to distant end 0x02 Flash Timing ON, inform host 0x03 Flash Timing ON, inform host and propagate
12	Current Layer 3 State
13	Host Service State 0x00 Out of Service 0x01 In Service
Signal Scanning Filters	
14, 15	Preseize
16, 17	Inseize
18, 19	Modified Incoming Release
20, 21	Modified Outgoing Release
22, 23	Normal Incoming Release
24, 25	Normal Incoming Release with Flash
26, 27	Normal Outgoing Release
28, 29	Normal Outgoing Release with Flash
30, 31	Post Inseize Acknowledge
32, 33	Outseize Acknowledge
34, 35	Outseize Answer
36, 37	Outseize Dial Signal End
38, 39	First Release
40, 41	ANI Request Off-hook
Signal Scanning Timers	
42, 43	Inseize Complete
44, 45	Post Inseize Complete
46, 47	Start Normal Outgoing Release
48, 49	Start Normal Incoming Release
50, 51	Outseize Acknowledge
52, 53	Outseize Answer
54, 55	Guard Timeout
56, 57	Release
58, 59	Glare Detection
60, 61	Minimum Flash
62, 63	Maximum Flash
64, 65	Start Dial Minimum Wink
66, 67	Start Dial Maximum Wink

Channel Parameter Query 0x0080

68, 69	Minimum Delay Dial Signal
70, 71	Maximum Delay Dial Signal
72, 73	Post Start Dial Signal Output Delay
74, 75	Wink 2 Min. Receive Duration
76, 77	Wink 2 Max. Receive Duration
78, 79	Wink 2 Min. Post Output Delay
80, 81	Wink 2 Max. Post Output Delay
82, 83	Wink 2 Max. Receive Detection
84, 85	Wink 3 Min. Receive Duration
86, 87	Wink 3 Max. Receive Duration
88, 89	Wink 3 Min. Post Output Delay
90, 91	Wink 3 Max. Post Output Delay
92, 93	Wink 3 Max. Receive Detection
94, 95	Wink 4 Min. Receive Duration
96, 97	Wink 4 Max. Receive Duration
98, 99	Wink 4 Min. Post Output Delay
100, 101	Wink 4 Max. Post Output Delay
102, 103	Wink 4 Max. Receive Detection
104, 105	Wink 5 Min. Receive Duration
106, 107	Wink 5 Max. Receive Duration
108, 109	Wink 5 Min. Post Output Delay
110, 111	Wink 5 Max. Post Output Delay
112, 113	Wink 5 Max. Receive Detection
114, 115	Wink 6 Min. Receive Duration
116, 117	Wink 6 Max. Receive Duration
118, 119	Wink 6 Min. Post Output Delay
120, 121	Wink 6 Max. Post Output Delay
122, 123	Wink 6 Max. Receive Detection
124, 125	Wink 7 Min. Receive Duration
126, 127	Wink 7 Max. Receive Duration
128, 129	Wink 7 Min. Post Output Delay
130, 131	Wink 7 Max. Post Output Delay
132, 133	Wink 7 Max. Receive Detection
134, 135	Wink 8 Min. Receive Duration
136, 137	Wink 8 Max. Receive Duration
138, 139	Wink 8 Min. Post Output Delay
140, 141	Wink 8 Max. Post Output Delay
142, 143	Wink 8 Max. Receive Detection
144, 145	Post ANI Off-Hook Output Delay
146, 147	Max. Receive ANI Off-Hook Request Detection
148, 149	Max. Receive Dialtone Detection
150, 151	MFR1 Min. Receive KP Duration
152, 153	MFR1 Min. Receive Digit Duration
154, 155	MFR1 Max. Receive KP Detection
156, 157	MFR1 Min. Receive Interdigit Duration
158, 159	MFR1 Max. Receive Interdigit Duration

Channel Parameter Query 0x0080

160, 161	DTMF Max. Receive 1st Digit Detection
162, 163	DTMF Min. Receive 1st Digit Detection
164, 165	DTMF Min. Receive Interdigit Duration
166, 167	DTMF Max. Receive Interdigit Duration
Transmit Signal Timers	
168, 169	Wink Duration
170, 171	Flash Duration
172, 173	Primary Outseize Signaling
174, 175	Outseize Signaling
176, 177	Delay Dial Signal Start
178, 179	Fixed Pause
180, 181	Timed Answer
182, 183	Ringing On Duration
184, 185	Ringing Off Duration
186, 187	Wink 1 Min. Transmit Delay
188, 189	Wink 2 Min. Transmit Delay
190, 191	Wink 2 Max. Transmit Delay
192, 193	Wink 2 Transmit Duration
194, 195	Wink 3 Min. Transmit Delay
196, 197	Wink 3 Max. Transmit Delay
198, 199	Wink 3 Transmit Duration
200, 201	Wink 4 Min. Transmit Delay
202, 203	Wink 4 Max. Transmit Delay
204, 205	Wink 4 Transmit Duration
206, 207	Wink 5 Min. Transmit Delay
208, 209	Wink 5 Max. Transmit Delay
210, 211	Wink 5 Transmit Duration
212, 213	Wink 6 Min. Transmit Delay
214, 215	Wink 6 Max. Transmit Delay
216, 217	Wink 6 Transmit Duration
218, 219	Wink 7 Min. Transmit Delay
220, 221	Wink 7 Max. Transmit Delay
222, 223	Wink 7 Transmit Duration
224, 225	Wink 8 Min. Transmit Delay
226, 227	Wink 8 Max. Transmit Delay
228, 229	Wink 8 Transmit Duration
230, 231	MFR1 Transmit KP Duration
232, 233	MFR1 Transmit Digit Duration
234, 235	MFR1 Transmit Interdigit Duration
236, 237	DTMF Transmit Digit Duration
238, 239	DTMF Transmit Interdigit Duration
Channel Configuration	

Channel Parameter Query 0x0080

240	Incoming Start Dial Type 0x00 Not Configured or Invalid for Trunk Type 0x01 Wink Start 0x02 Delay Dial 0x03 Immediate 0x04 Dial Tone
241	Outgoing Start Dial Type 0x00 Not Configured or Invalid for Trunk Type 0x01 Wink Start 0x02 Delay Dial 0x03 Fixed Pause 0x04 Dial Tone
242	Busy Out Flag 0x00 Busy Out disabled on this trunk 0x01 Busy Out enabled on this trunk
243	Distant End Release Mode 0x01 Park 0x02 Release
244	Answer Supervision Mode 0x00 Propagate Answer To The Distant End 0x01 Notify Host Of Answer 0x02 Propagate Answer To The Distant End And Notify Host Of Answer 0x03 No Answer Supervision (no propagation of answer or notification)
245	Local End Release Mode 0x01 Park 0x02 Release
246	PCM Encoding Format 0x01 μ -law Encoding 0x02 A-law Encoding
247	Checksum

Channel Release Request 0x0037

SwitchKit Name	ChannelReleaseRequest
Type	EXS API and SwitchKit API message
Description	<p>Channel Release Request 0x0037</p> <p>The conditions in which the CSP uses this message depends on the protocol you are using.</p> <p>SS7 An SS7 channel reports this message to the host if both of the following conditions exist:</p> <ul style="list-style-type: none"> • The SS7 channel is connected to a channel that is released • The release modes for the channels call for the SS7 channel to be released <p>This lets the host insert parameters for the outgoing ISUP REL using the <i>Release Channel With Data</i> message. If no data is to be inserted, the host may use the <i>Release Channel</i> message.</p> <p>ISDN The CSP sends this message to the host upon a network-initiated <i>ISDN Disconnect</i> message if the ISUP mode configuration is set to allow release upon a disconnect. The host responds with either a <i>Release Channel</i> message or a <i>Release Channel With Data</i> message to send information to the ISDN with the release.</p> <p>Sent by CSP</p>

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short Span;
    UBYTE Channel;
    UBYTE ReleaseDataType;
    UBYTE Data[222];
} XL_ChannelReleaseRequest;
```

C++ Class

```
class XLC_ChannelReleaseRequest : public
    XLC_OneChannelMessage {
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getReleaseDataType() const;
    void setReleaseDataType(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0005)
3, 4	Message Type (0x0037)	3, 4	Message Type (0x0037)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB 0x00 - Individual AEs Number of AEs to follow AEs 0x0D Channel	8	Checksum
:	Release Data Type 0x00 No Data 0x03 ISDN ICB 0x04 SS7 ICB		
:	Data ICBs The data format depends on the Release Data Type. 0x10 ISDN Formatted IEs 0x12 SS7 Parameters 0x25 ISDN Segmented Message		
	Refer to <i>Interworking TLVs (4-4)</i> if you have enabled interworking from the <i>VoIP Protocol Configure 0x00EE</i> message.		
:	Checksum		

Channel Released 0x0049

SwitchKit Name ChannelReleased

Type EXS API and SwitchKit API message

Description: **Channel Released 0x0049**

This message informs the host that a channel has been released, is in an idle state, and is ready for incoming or outgoing calls.

Sent by: CSP

Resent in EXS API This message is resent once after five seconds.

SwitchKit Code **C Structure:**

```
typedef struct {
    unsigned short Span;
    UBYTE Channel;
} XL_ChannelReleased;
```

C++ Class:

```
class XLC_ChannelReleased : public XLC_OneChannelMessage
{
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0005)
3, 4	Message Type (0x0049)	3, 4	Message Type (0x0049)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u> Address Method 0x00 - Individual AEs Number of AEs to follow AE 0x0D Channel	8	Checksum
:	Checksum		

Channel Released With Data 0x0069

SwitchKit Name ChannelReleasedWithData

Type EXS API and SwitchKit API message

Description **Channel Released With Data 0x0069**

This message indicates that the network has initiated a release with associated data. If no parameter data is associated with the call release, then a *Channel Released* (0x49) message is sent instead.

For a VoIP call, a Release TLV will appear in the message in the Extended ICB. For SS7 TUP, Release parameters are included in a BT IUP parameters ICB. When BT IUP is configured, then a BT IUP ICB with a release TLV may be present.

For SwitchKit Only

When an ICBType field is set to 0x03, use the following methods for extracting the data of the fields shown in the following table.

If the field is...	then the method for extracting the data is...
ICBSubType	sk_extractExtendedICBFromChannelReleasedWithData()
ICBLength	sk_extractExtendedICBFromChannelReleasedWithData()
ICBData	sk_extractExtendedICBFromChannelReleasedWithData()
getICBSubType()	getExtendedICBSubtype()
getICBLength()	getExtendedICBLength()
getICBData()	getExtendedICBData()

Sent by CSP

Resent in EXS API This message is resent once after five seconds.

**Example Message
(Log Output in SwitchKit)**

The following socket log output/example message is an H.323 call, that the network has initiated a release with a Release TLV indicating Invalid Permission Chosen, Facility Call Deflection.

```
00 5d 00 69 00 2c 01 00 01 0d 03 00 00 04
02 02 1e 2a 00 05 01 04 00 04 00 00 00 00
01 05 00 04 00 00 00 00 01 11 00 04 00 00
00 00 01 10 00 04 00 00 00 00 01 12 00 04
00 00 00 00 03 00 33 00 1e 00 04 27 4e 00
02 00 10 27 92 00 04 c0 a8 00 1a 27 93 00
04 00 00 20 5c 27 e3 00 02 02 0c
```

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short Span;
```

```

    UBYTE Channel;
    UBYTE ICBCount;
    UBYTE ICBType;
    UBYTE ICBSubtype;
    UBYTE ICBLength;
    UBYTE ICBDData[219];
} XL_ChannelReleasedWithData;
DllExport bool
sk_extractExtendedICBFromChannelReleasedWithData(in
t* xtICBSubtype, int* xtICBLength, UBYTE** ICBDData,
XL_ChannelReleasedWithData* crwd)

```

C++ Class

```

class XLC_ChannelReleasedWithData : public
    XLC_OneChannelMessage {
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getICBCount() const;
    void setICBCount(UBYTE x);
    UBYTE getICBType() const;
    void setICBType(UBYTE x);
    UBYTE getICBSubtype() const;
    void setICBSubtype(UBYTE x);
    UBYTE getICBLength() const;
    void setICBLength(UBYTE x);
    const UBYTE *getICBDData() const;
    UBYTE *getICBDData();
    void setICBDData(UBYTE *x);
    XBYTE getExtendedICBSubtype() const;
    XBYTE getExtendedICBLength() const;
    const UBYTE *getExtendedICBDData() const;
    UBYTE *getExtendedICBDData();
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0005)
3, 4	Message Type (0x0069)	3, 4	Message Type (0x0069)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u> Address Method 0x00 - Individual AEs Number of AEs to follow AE 0x0D Channel	8	Checksum

:	Number of ICBs to follow (Ignore this field if no ICBs in message.)
:	ICB 0x02 Data Type 0x12 SS7 Parameters 0x1C SS7 TUP Formatted Fields 0x1E Generic PPL in Channel Released with Data (See ICB format below this table) 0x1F SS7 Unformatted Raw Parameters 0x23 BT IUP Parameters 0x03 Extended Data Type 0x0033 NPDI Universal ICB in Channel Released with Data (See ICB format below this table)
	Refer to <i>Interworking TLVs (4-4)</i> if you have enabled interworking from the <i>VoIP Protocol Configure 0x00EE</i> message.
:	Checksum

0x1E Generic PPL in *Channel Released with Data*

When an IP channel is released, the CSP sends a *Channel Released with Data* message to the host. The message contains the following data, providing IP statistics for the IP call:

ICB Type	Data (0x02)
ICB ID	Generic PPL ICB (0x1E)
Data Length	0x2A
Number of TLVs	0x0005
Tag	0x0104 RTP Packets Lost
Length	0x0004
Value (4 Bytes)	Data[0-3] Number of Packets
Tag	RTP Packets Received (0x0105)
Length	0x0004
Value (4 Bytes)	Data[0-3] Number of Packets
Tag	RTP Packets Sent (0x0111)
Length	0x0004
Value (4 Bytes)	Data[0-3] Number of Packets
Tag	RTP Octets Received (0x0110)
Length	0x0004
Value (4 Bytes)	Data[0-3] Number of Octets
Tag	RTP Octets Sent (0x0112)
Length	0x0004
Value (4 Bytes)	Data[0-3] Number of Octets

0x0033 NPD I Universal ICB in *Channel Released with Data*

ICB Type	Data (0x03)
ICB ID	NPDI (0x0033)
Data Length	Variable
Number of TLVs	0x0004
Tag	RTP IP (0x2792)
Length	0x0004
Value (4 Bytes)	Variable
Tag	RTP Port (0x2793)
Length	0x0004
Value (4 Bytes)	Variable
Tag	NPDI Message Type (0x274E)
Length	0x0002
Value (2 Bytes)	Variable
Tag	Release Cause Code (0x27E3) For H.323
Length	0x0002
Value (2 Bytes)	Variable

ChannelGroupContentsQuery

Type: EXS SwitchKit API message

Description Use the *SK_ChannelGroupContentsQuery* to query the contents of the channel group by name. The acknowledgment *SK_ChannelGroupContentsQueryAck* contains the number of channels in that group as well as all the spans and channels.

To process all these channels, you use the following code:

```
SK_MSG_SWITCH(msg) {
    CASE_ChannelGroupContentsQueryAck(ack) {
        int i;
        UBYTE *dta = ack->Channels;
        for (i=0;i<ack->NumChannels;i++)
        {
            short span = ntohs(*(short *)dta);
            int chan = (int)dta[2];
            processChannel(span,chan);
            dta += 3;
        }
    }
} SK_END_SWITCH;
```

The format of the Channels data structure is two bytes for the span in network byte order, followed by one byte representing the channel offset.

Sent by Application

C Structure

```
typedef struct {
    char GroupName[50];
    UBYTE reserved67[23];
} SK_ChannelGroupContentsQuery;
```

C Structure Response

```
typedef struct {
    int Status;
    int NumChannels;
    UBYTE Channels[245];
} SK_ChannelGroupContentsQueryAck;
```

C++ Class

```
class SKC_ChannelGroupContentsQuery : public
    SK_ToolkitMessage {
public:
    const char *getGroupName() const;
    void setGroupName(const char *x);
};
```

C++ Class Response

```
class SKC_ChannelGroupContentsQueryAck : public
    SKC_ToolkitAck {
public:
    int getStatus() const;
    void setStatus(int x);
    int getNumChannels() const;
    void setNumChannels(int x);
    const UBYTE *getChannels() const;
    UBYTE *getChannels();
    void setChannels(UBYTE *x);
};
```

Status Field Return Values

The possible status values are:

OK

Query successful acknowledged.

SK_INVALID_GROUP

The GroupName in the query does not exist.

Arguments

GroupName: Name of Group to be queried.

ChannelGroupPopulationQuery

Type:	EXS SwitchKit API message
Description	Use the <i>SK_ChannelGroupPopulationQuery</i> message to query the existing defined channel groups on the LLC. The GroupNames field contains a set of group names, each separated by the NULL character. For example, to loop over all the groups, as returned by this message, you could use the following code:
	<pre>SK_MSG_SWITCH(msg) { CASE_ChannelGroupPopulationQueryAck(ack) { int i; char *gps = (char *)ack->GroupNames; for (i=0;i<ack->NumGroups;i++) { char *aGroup = gps; processGroup(aGroup); gps += strlen(gps) + 1; } } } SK_END_SWITCH;</pre>
Sent by	Application
C Structure	<pre>typedef struct { } SK_ChannelGroupPopulationQuery;</pre>
C Structure Response	<pre>typedef struct { int Status; int NumGroups; UBYTE GroupNames[245]; } SK_ChannelGroupPopulationQueryAck;</pre>
C++ Class	<pre>class SKC_ChannelGroupPopulationQuery : public SKC_ToolkitMessage { public: };</pre>
C++ Class Response	<pre>class SKC_ChannelGroupPopulationQueryAck : public SKC_ToolkitAck { public: int getStatus() const; void setStatus(int x); int getNumGroups() const; void setNumGroups(int x); const UBYTE *getGroupNames() const; UBYTE *getGroupNames(); void setGroupNames(UBYTE *x); };</pre>

ChannelGroupPopulationQuery

Return Status Always 0x10

ChannelProblem

Type:	EXS SwitchKit API message
Description	<p>This message is generated by SwitchManager. To do this, SwitchManager must be running in order for an application to receive a channel problem message. An application can receive a message when a channel has encountered some problems, but has not gone out-of-service. The application also receives this message when the problem clears.</p> <p>ErrorNum equals one (1) if a red or yellow span alarm occurs; or equals two (2) if a PPL Event Indication occurs. The LLC does not allocate a channel that is in an alarm state, but if an application already has a channel that has entered into an alarm state, the application must deal with this state. Text will contain a description of the problem, which in the case of red and yellow alarms is a description of the current network status. A problem with a value of one (1) means the problem just occurred; a value of zero (0) means the problem has cleared. In the case of a span alarm, Data1 will equal the status field extracted from the EXS API message. In the case of a PPL Event Indication, Data1 will equal the PPL Event.</p>
Sent by	SwitchManager; forwarded by LLC
C Structure	<pre>typedef struct { unsigned short Span; UBYTE Channel; UBYTE Problem; UBYTE ErrorNum; UBYTE Data1; UBYTE Data2; char Text[200]; } SK_ChannelProblem;</pre>
C++ Class	<pre>class SKC_ChannelProblem : public SKC_ToolkitMessage { public: unsigned short getSpan() const; void setSpan(unsigned short x); UBYTE getChannel() const; void setChannel(UBYTE x); UBYTE getProblem() const; void setProblem(UBYTE x);} UBYTE getErrorNum() const; void setErrorNum(UBYTE x); UBYTE getData1() const; void setData1(UBYTE x);</pre>

ChannelProblem

```
UBYTE getData2() const;  
void setData2(UBYTE x);  
const char *getText() const;  
void setText(const char *x);  
};
```

Clear System Software 0x000C

SwitchKit Name	ClearSystemSoftware
Type	EXS API and SwitchKit API message
Description	Clear System Software 0x000C

NOTE: This message is for use in a **lab environment only**. You must not send this message to a live system.

This message clears the system software stored in memory on the CSP Matrix Series 3 Card. Use this message immediately before you send a new load to the CSP Matrix Series 3 Card. To stop the running system software, push the Reset button or send a *Reset Matrix* message.

System software is copied into maintained storage. Upon reset or boot, software is copied from maintained storage to RAM for execution. When the host sends this message, the copy in maintained storage is cleared, but the RAM is unaffected.

If a card resets after you send this message, the card will not retrieve a software load from maintained storage and will not come back into service.

Redundant Matrix Cards

In a redundant system, sending this message to the active CSP Matrix Series 3 Card clears both CSP Matrix Series 3 Card. After sending this message to the active CSP Matrix Series 3 Card, you must send a *Reset Matrix* message to **both** CSP Matrix Series 3 Card.

Sent by Host

Related API Message *Reset Matrix* 0x009D (ResetMatrix)

SwitchKit Code **C Structure**

```
typedef struct {
    } XL_ClearSystemSoftware;
```

C++ Class

```
class XLC_ClearSystemSoftware : public
    XLC_OutboundMessage {
public:
    };
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x0005)	1, 2	Length (0x0007)
3, 4	Message Type (0x000C)	3, 4	Message Type (0x000C)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Checksum	8, 9	Status (MSB, LSB)
		10	Checksum

ClearAllChanGroups

Type:	EXS SwitchKit API message
Description	Use the <i>SK_ClearAllChanGroups</i> message to remove all channel groups.
Sent by	SwitchManager
Configuration	<i>ClearAllChanGroups</i> ();
C Structure	<pre>typedef struct { } <i>SK_ClearAllChanGroups</i> ;</pre>
C++ Class	<pre>class <i>SKC_ClearAllChanGroups</i> : public SKC_AdminMessage{ public: };</pre>

ClearChanGroup

Type:	SwitchKit API message
Description	Use the <i>SK_ClearChanGroup</i> message to remove all channels from a channel group. Because channel group bindings accumulate for as long as the LLC is running, it is recommended that you clear every channel group that you use in a configuration file.
Sent by	SwitchManager
Configuration	<i>ClearChanGroup</i> (ChannelGroup = string);
C Structure	typedef struct { char ChannelGroup[45]; } <i>SK_ClearChanGroup</i> ;
C++ Class	class <i>SKC_ClearChanGroup</i> : public SKC_AdminMessage { public: const char *getChannelGroup() const; void setChannelGroup(const char *x); };

ClearLog

Type	SwitchKit API message
Description	The <i>ClearLog</i> message is used as an operations, administration and maintenance feature in the Converged Services Administrator (CSA) to clear log files. You can choose among different options to clear log files.
Sent by	Application or CSA
How to use	To use a <i>ClearLog</i> message, construct the message and set the LoggingBitMask: <pre>SKC_ClearLog cl; UBYTE log_bit_mask = SK_CLEAR_MSG_LOG SK_CLEAR_SOCKET_LOG; cl.setLoggingBitMask(log_bit_mask);</pre>
Argument Values	The following table shows the common values of the LoggingBitMask argument. You can combine two or more values with a logical OR to clear several logs at once.

Clear individual log files using the following values:

Argument Value	Description
SK_CLEAR_MSG_LOG	Clears only the messages log file.
SK_CLEAR_MAINT_LLC_LOG	Clears only the LLC maintenance log.
SK_CLEAR_SOCKET_LOG	Clears only the socket log file.
SK_CLEAR_ALARM_LOG	Clears only the alarm log file.
SK_CLEAR_MAINT_SWMGR_LOG	Clears only the SwitchManager maintenance log file.

Clear groups of log files using the following values:

Argument Value	Description
SK_CLEAR_ALL_SK_LOGS	Clears all the SwitchKit logs.
SK_CLEAR_ALL_LLC_LOGS	Clears all LLC logs, including: SK_CLEAR_MSG_LOG SK_CLEAR_MAINT_LLC_LOG SK_CLEAR_SOCKET_LOG

SK_CLEAR_ALL_SWMGR_LOGS	Clears all SwitchManager logs, including: SK_CLEAR_ALARM_LOG SK_CLEAR_MAINT_SWMGR_LOG
-------------------------	--

C Structure typedef struct {
 UBYTE LoggingBitMask;
} *SK_ClearLog*;

C Structure Response typedef struct {
 int Status;
} *SK_ClearLogAck*;

C++ Class class *SKC_ClearLog* : public SKC_ToolkitMessage {
public:
 UBYTE getLoggingBitMask() const;
 void setLoggingBitMask(UBYTE x);
};

C++ Class Response class *SKC_ClearLogAck* : public SKC_ToolkitAck {
public:
 int getStatus() const;
 void setStatus(int x);
};

Clear IP Signaling Series 3 Card 0x0108

Type EXS API message

Description **Clear IP Signaling Series 3 Card 0x0108**

This message clears the system software on the IP Signaling Series 3 card before and/or after the IP Signaling Series 3 card interface is established. This message is sent directly to the IP Signaling Series 3 card and uses the extended logical node ID format.

To initiate the download of the new software load, you need to reset the IP Signaling Series 3 card by pushing the Reset button or using the *Reset IP Signaling Series 3 Card* message (0x0107).

Sent by Host

Example Message The following clears the system software on the IP Signaling Series 3 card when the IP Signaling Series 3 card ID or Matrix ID is not assigned.

00 11 01 08 00 00 fe 00 02 52 02 ff ff 53 04 00 20 ff ff

Hex API Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x0108)	3, 4	Message Type (0x0108)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Sequence Number (same as message)
7	Extended Logical Node ID (0xFE)	7	Extended Logical Node ID (0xFE)
:	<u>AIB</u> Address Method 0x00 - Individual AEs Number of AEs to follow AEs 0x48 Ring Communication Link 0x53 Object Type	8, 9	Status MSB, LSB (0x0010)
:	Checksum	:	<u>AIB</u> Same as message
		:	Checksum

ClearSoftware

- Type:** SwitchKit API message
- Purpose** Use the *ClearSoftware* message to clear system software and to reset the CSP Matrix Series 3 Card or device server of the CSP.
- Description** This message is used for clearing system software and for resetting the the CSP Matrix Series 3 Card(s).
- Sent by** Application or CSA
- How to use** To use a *ClearSoftware* message, construct the message and set the matrix and action arguments as shown below:
- ```
SKC_ClearSoftware cls();
cls.setMatrix(matrix);
cls.setAction(action);
cls.setNodeId(node);
```
- Arguments** The following table shows the arguments you can change:

| Arguments | Description                                                                                                                                                                                                                                                                                                                                  |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Matrix    | <p>This argument specifies on which CSP Matrix Series 3 Card the action is applied:</p> <p>SK_CLEARSW_ACTIVE (1) = Apply action to active CSP Matrix Series 3 Card</p> <p>SK_CLEARSW_STANDBY (2) = Apply action to standby CSP Matrix Series 3 Card</p> <p>SK_CLEARSW_ALL (3) = Apply action on both CSP Matrix Series 3 Card in the CSP</p> |
| Action    | <p>This argument specifies which action is applied:</p> <p>SK_CLEARSW_CLEAR (1) = Clear software but do not reset the CSP Matrix Series 3 Card.</p> <p>SK_CLEARSW_CLEAR_AND_RESET (2) = Clear software and reset the CSP Matrix Series 3 Card.</p>                                                                                           |

| <b>Arguments</b> | <b>Description</b>                                                                                                                                                       |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Node             | This argument specifies the logical node ID of the CSP Matrix Series 3 Card pair or the Device Server ID of the Device Server card to which the action is to be applied. |

**Status Field** The following table shows the possible return values of the status field. The status field can contain any of the values or the logical OR combination of two or more of the following:

| Value                           | Description                                                                     |
|---------------------------------|---------------------------------------------------------------------------------|
| SK_SUCCESS (0x10)               | Message successfully executed.                                                  |
| SK_ACTIVE_NOT_CLEARED (0x0001)  | The active CSP Matrix Series 3 Card was not cleared.                            |
| SK_STANDBY_NOT_CLEARED (0x0002) | The standby CSP Matrix Series 3 Card was not cleared.                           |
| SK_ACTIVE_NOT_RESET (0x0004)    | The active CSP Matrix Series 3 Card was not reset.                              |
| SK_STANDBY_NOT_RESET (0x0008)   | The standby CSP Matrix Series 3 Card was not reset.                             |
| SK_INVALID_NODE (0xf002)        | This indicates that the application has no description data associated with it. |

**C Structure**

```
typedef struct {
 UBYTE Matrix;
 UBYTE Action;
 unsigned short NodeId;
} SK_ClearSoftware;
```

**C++ Class**

```
class SKC_ClearSoftware : public SKC_ToolkitMessage {
public:
 UBYTE getMatrix() const;
 void setMatrix(UBYTE x);
 UBYTE getAction() const;
 void setAction(UBYTE x);
 unsigned short getNodeid() const;
 void setNodeid(unsigned short x);
};
```

## Collect Digit String 0x00BC

---

**SwitchKit Name** CollectDigitString

**Type** EXS API and SwitchKit API message

**Description** **Collect Digit String 0x00BC**

This message instructs the CSP to attach a digit receiver to the specified channel.

**NOTE:** The digit receiver collects digit strings only if the digits are either DTMF or MFR1, or if the call is in a state other than idle, out of service, or busied out

Once the receiver has collected the digits, the CSP sends a *Call Processing Event* message with the value 0x02 (Digits) in the *Call Processing Event* field. If an error occurs, the CSP sends a *Call Processing Event* message and indicates the error using the *Status* field.

If the host sends this message and specifies a channel to which a digit receiver is already attached, the CSP sends the host a response with a status of 0x30 (DSP Resource Already Allocated). The channel is not affected.

The CSP releases the digit receiver from the channel when one of the following occurs:

- The CSP sends a *Call Processing Event* message
- The call is released
- The CSP receives a *DSP Service Cancel* message from the host

**Sent by** Host Application

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
 UBYTE Mode;
 UBYTE MaxDigits;
 UBYTE NumTermChars;
 UBYTE ConfigBits;
 unsigned short TermChars;
 unsigned short InterDigitTimer;
 unsigned short FirstDigitTimer;
 unsigned short CompletionTimer;
 unsigned short MinReceiveDigitDuration;
 UBYTE AddressSignallingType;
 UBYTE NumDigitStrings;
 unsigned short ResumeDigitCltnTimer;
} XL_CollectDigitString;
```

### C++ Class

```
class XLC_CollectDigitString : public
 XLC_OneChannelOutbound {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 UBYTE getMode() const;
 void setMode(UBYTE x);
 UBYTE getMaxDigits() const;
 void setMaxDigits(UBYTE x);
 UBYTE getNumTermChars() const;
 void setNumTermChars(UBYTE x);
 UBYTE getConfigBits() const;
 void setConfigBits(UBYTE x);
 unsigned short getTermChars() const;
 void setTermChars(unsigned short x);
 unsigned short getInterDigitTimer() const;
 void setInterDigitTimer(unsigned short x);
 unsigned short getFirstDigitTimer() const;
 void setFirstDigitTimer(unsigned short x);
 unsigned short getCompletionTimer() const;
 void setCompletionTimer(unsigned short x);
 unsigned short getMinReceiveDigitDuration() const;
 void setMinReceiveDigitDuration(unsigned short x);
 UBYTE getAddressSignallingType() const;
 void setAddressSignallingType(UBYTE x);
 UBYTE getNumDigitStrings() const;
 void setNumDigitStrings(UBYTE x);
 unsigned short getResumeDigitCltnTimer() const;
 void setResumeDigitCltnTimer(unsigned short x);
};
```

**Overview of message** The following table provides an overview of this message. The table following it provides the detail for each byte.

| MESSAGE (White) |                                                 | RESPONSE (Gray) |                                                          |
|-----------------|-------------------------------------------------|-----------------|----------------------------------------------------------|
| Byte            | Field Description                               | Byte            | Field Description                                        |
| 0               | Frame (0xFE)                                    | 0               | Frame (0xFE)                                             |
| 1, 2            | Length (0xNNNN)                                 | 1, 2            | Length (0x0005)                                          |
| 3, 4            | Message Type (0x00BC)                           | 3, 4            | Message Type (0x00BC)                                    |
| 5               | Reserved (0x00)                                 | 5               | Reserved (0x00)                                          |
| 6               | Sequence Number                                 | 6               | Same Sequence Number                                     |
| 7               | Logical Node ID                                 | 7               | Logical Node ID                                          |
| :               | AIB (Starting with byte 0)                      | 8, 9            | Status (MSB, LSB)<br>0x0002 DSP Receiver Request Timeout |
|                 |                                                 | 10              | Checksum                                                 |
| :               | Mode                                            |                 |                                                          |
| :               | Max Digits                                      |                 |                                                          |
| :               | Number of Terminating Characters                |                 |                                                          |
| :               | Configuration Bits                              |                 |                                                          |
| :               | Terminating Characters (MSB, LSB)               |                 |                                                          |
| :               | Interdigit Timer (MSB, LSB)                     |                 |                                                          |
| :               | First Digit Timer (MSB, LSB)                    |                 |                                                          |
| :               | Completion Timer (MSB, LSB)                     |                 |                                                          |
| :               | Minimum Receive Digit Duration Timer (MSB, LSB) |                 |                                                          |
| :               | Address Signaling Type                          |                 |                                                          |
| :               | Number of Digit Strings                         |                 |                                                          |
| :               | Resume Digit Collection Timer (MSB, LSB)        |                 |                                                          |
| :               | Checksum                                        |                 |                                                          |

**Hex API Format - Detailed**

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | RESPONSE (Gray) |                       |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 0               | Frame (0xFE)          |
| 1, 2            | Length 0xNNNN                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 1, 2            | Length 0xNNNN         |
| 3, 4            | Message Type (0x00BC)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 3, 4            | Message Type (0x00BC) |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 7               | Logical Node ID       |
| :               | AIB                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 8, 9            | Status MSB, LSB       |
|                 | Address Method<br>0x00 - Individual AEs                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 10              | Checksum              |
|                 | Number of AEs to follow                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                 |                       |
|                 | AE<br>0x0D Channel                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                 |                       |
| :               | <p>Mode</p> <p>0x01 Fixed Number of Digits</p> <p>0x02 Use Terminating Characters</p> <p>0x03 Use KP_ST Characters</p> <p>0x04 Use Fixed Number of Digits or Terminating Characters</p> <p>0x05 Reserved</p> <p>0x06 Reserved</p> <p>0x07 Reserved</p> <p>0x08 Fixed or Indefinite Number of Digits</p> <p>If the first digit detected is zero, the Indefinite Number of Digits method is used and the Completion Timer or Interdigit Timer is used (whichever expires first) to terminate digit collection. If the first digit collected is not zero, the Fixed Number of Digits method is used. This method is used to help facilitate collecting digit strings on channels that process national and international calls.</p> |                 |                       |
| :               | <p>Max Digits</p> <p>Maximum number of digits to collect (0–100)</p> <p>This field is valid only if the Mode field value is either 0x01 (Fixed Number of Digits) or 0x04 (Use Fixed Number of Digits or Term Chars); otherwise, it is ignored.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                 |                       |
| :               | <p>Number of Terminating Characters</p> <p>Number of terminating characters to collect (0–4).</p> <p>This field is valid only if the Mode field value is either 0x02 (Use Terminating Characters) or 0x04 (Use Fixed Number of Digits or Term Chars); otherwise, it must be 0x00.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                            |                 |                       |

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | <p>Configuration Bits</p> <p>Bit</p> <p><b>0 Report Digit at Falling Edge</b></p> <p>0 Digit is reported at its rising edge when the digit is first pressed or if it is present on the line at the time the receiver is attached.</p> <p>1 Digit is reported at its falling edge when the digit is first released.</p> <p><b>1 First Digit Collected</b></p> <p>0 Do not inform the host that the first digit was collected.</p> <p>1 Inform the host that the first digit was collected. Use the Call Processing Event message. The first digit will also be reported in the Call Processing Event message with the remaining digits.</p> <p><b>2 Ignore # Character</b></p> <p>0 Do not ignore the “#” character as the first digit.</p> <p>1 Ignore the “#” character as the first digit.</p> <p><b>3 Suspend Digit Collection</b></p> <p>0 Do not suspend digit collection.</p> <p>1 Suspend digit collection upon outputting digits.</p> <p>The following requires that one DSP-2, DSP-ONE or MFDSP card be configured with both DTMF Transmission and DTMF Reception. Channels with this feature enabled will only have access to DTMF Receivers on cards which also have DTMF Transmitters configured.</p> <p><b>4 Cancel Prompting Tone or Recorded Announcement</b></p> <p>0 Prompting tone or RAN is cancelled on receipt of the first digit.</p> <p>1 Prompting tone or recorded announcement continues.</p> <p><b>5 Detect Leading Silence (only used for rising edge detection)</b></p> <p>0 Do not require leading silence before reporting valid rising edge digit.</p> <p>1 Require detection of leading silence before reporting the first valid rising edge digit. Digits on the line at time of receiver attachment will not be reported.</p> <p><b>6 Cancel Recording</b></p> <p>0 Recording is cancelled on receipt of the first digit</p> <p>1 Recording continues</p> <p><b>7 Detect DTMF Frequencies</b></p> <p>0 DTMF detection logic checks the two individual DTMF frequencies and the total of both frequencies that make up a DTMFdigit before reporting the detected digit.</p> <p>1 DTMF detection logic only checks the total frequency of the two individual DTMF frequencies that make up a DTMFdigit before reporting the detected digit.<br/>(Default)</p> |
|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Collect Digit String 0x00BC

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | <p>Terminating Characters MSB, LSB</p> <p>Use this two-byte field only if the value of the Mode field is either 0x02 (Use Terminating Characters) or 0x04 (Use Fixed Number of Digits or Term Chars). The terminating characters are stored as nibbles. You can define up to four terminating characters, as shown below.</p> <p>MSB</p> <p>Bit</p> <p>0–3 Terminating Character 2</p> <p>4–7 Terminating Character 1</p> <p>LSB</p> <p>Bit</p> <p>0–3 Terminating Character 4</p> <p>4–7 Terminating Character 3</p> |
| : | <p>Interdigit Timer MSB, LSB</p> <p>NOTE: Timers are in 10 ms units.</p> <p>The maximum time to wait between digits after the first one is received.</p> <p>The value 0xFFFF disables this timer.</p>                                                                                                                                                                                                                                                                                                                 |
| : | <p>First Digit Timer MSB, LSB</p> <p>The maximum time to wait for the first digit of a string (in 10 ms units).</p> <p>The value 0xFFFF disables this timer.</p>                                                                                                                                                                                                                                                                                                                                                      |
| : | <p>Completion Timer MSB, LSB</p> <p>The maximum time to wait for the entire string after the first digit is detected.</p> <p>The value 0xFFFF disables this timer.</p>                                                                                                                                                                                                                                                                                                                                                |
| : | <p>Minimum Receive Digit Duration Timer MSB, LSB</p> <p>The minimum time a digit must be detected before it is declared valid.</p> <p>The value must be at least 30 ms, which is the default.</p>                                                                                                                                                                                                                                                                                                                     |
| : | <p>Address Signaling Type</p> <p>The address signaling type must be the same for both strings within a stage, but may differ between stages.</p> <p>0x01 DTMF</p> <p>0x02 MFR1</p> <p>0x14 DTMF and Dial Pulse Receiver</p> <p>0x15 Dial Pulse Receiver</p>                                                                                                                                                                                                                                                           |
| : | <p>Number of Digit Strings</p> <p>Number of digit strings to collect (1 or 2, usually 1 unless strings are KP/ST framed)</p>                                                                                                                                                                                                                                                                                                                                                                                          |
| : | <p>Resume Digit Collection Timer MSB, LSB</p> <p>The amount of time to wait after outpulsing is complete to resume digit collection.</p>                                                                                                                                                                                                                                                                                                                                                                              |
| : | <p>Checksum</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

# ConfigChanGroup

---

|                      |                                                                                                                                                                                                                                                                               |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Type:</b>         | SwitchKit API message                                                                                                                                                                                                                                                         |
| <b>Description</b>   | Use the <i>SKC_ConfigChanGroup</i> message to configure parameters for a channel group.                                                                                                                                                                                       |
| <b>Sent by</b>       | SwitchManager                                                                                                                                                                                                                                                                 |
| <b>Configuration</b> | <i>ConfigChanGroup</i> (<br>ChannelGroup = string,<br>Entity = integer,<br>Parameter = integer);                                                                                                                                                                              |
| <b>C Structure</b>   | typedef struct {<br>UBYTE Entity;<br>UBYTE Parameter;<br>char GroupName[50];<br>} <i>SK_ConfigChanGroup</i> ;                                                                                                                                                                 |
| <b>C++ Class</b>     | class <i>SKC_ConfigChanGroup</i> : public SKC_AdminMessage {<br>public:<br>UBYTE getEntity() const;<br>void setEntity(UBYTE x);<br>UBYTE getParameter() const;<br>void setParameter(UBYTE x);<br>const char *getGroupName() const;<br>void setGroupName(const char *x);<br>}; |

## Entity Options    Entity

### 0 - Allocation Pattern

Controls the order in which LLC will allocate the channels when you request a channel in a channel group.

### Parameters for Allocation Pattern

| Value     | Allocator   | Description                                                                                                                                                      |
|-----------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 default | Round Robin | Begins with the first available channel, and proceeds through the group, always choosing the next available channel starting from the channel it last allocated. |
| 1         | Ascending   | Always chooses the first (i.e., lowest numbered) available channel in the group.                                                                                 |

## ConfigChanGroup

| <b>Value</b> | <b>Allocator</b>          | <b>Description</b>                                                         |
|--------------|---------------------------|----------------------------------------------------------------------------|
| 2            | Descending                | Chooses the last (highest numbered) available channel in the group.        |
| 3            | LRU (Least Recently Used) | Chooses the channel that has been in the idle state for the longest time.  |
| 4            | MRU (Most Recently Used)  | Chooses the channel that has been in the idle state for the shortest time. |

**Example** *ConfigChanGroup* (  
    ChannelGroup = "outbound",  
    Entity = 0,  
    Parameter = 1);

# ConfigStatusMsg

---

|                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Type:</b>        | SwitchKit API message                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Purpose</b>      | <p>SwitchManager sends the <i>SK_ConfigStatusMsg</i> message to a registered application at configuration time.</p> <p>An application can register to receive this message when SwitchManager configures the entire CSP or a part of an CSP configuration, including start-up, a dynamic change request, or a matrix failure. SwitchManager tracks when the configuration begins, when it ends, and the number of positive and negative ACKs received.</p> <p>To receive the message <i>SK_ConfigStatusMsg</i>, your application must register for this message using the <code>sk_msgRegister()</code> function. The application should not send an acknowledgement for this message.</p> <p>If SwitchManager is run with <code>-s</code> option and if there was no configuration message sent to the CSP, SwitchManager will not send any <i>SK_ConfigStatusMsg</i>.</p> |
| <b>Sent by</b>      | SwitchManager                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>C Structure</b>  | <pre>typedef struct {     UBYTE Event;     UBYTE ConfigType;     int NumMsgsSent;     int NumMsgsNacked; } <i>SK_ConfigStatusMsg</i>;</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>C++ Class</b>    | <pre>class <i>SKC_ConfigStatusMsg</i> : public SKC_ToolkitMessage { public:     UBYTE getEvent() const;     void setEvent(UBYTE x);     UBYTE getConfigType() const;     void setConfigType(UBYTE x);     int getNumMsgsSent() const;     void setNumMsgsSent(int x);     int getNumMsgsNacked() const;     void setNumMsgsNacked(int x); };</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>Field Values</b> | The following table shows the possible field values of this message:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

| Field         | Values and Description                                                                                                                                                                                                                                                                                                                                                              |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Event         | <ul style="list-style-type: none"> <li>• SK_BEGIN_CONFIG = indicates SwitchManager began a configuration.</li> <li>• SK_END_CONFIG = indicates SwitchManager finished the configuration.</li> <li>• SK_PARSE_ERROR = indicates an error occurred when parsing the configuration.</li> </ul>                                                                                         |
| ConfigType    | <p>This field indicates the reason the configuration takes place. Possible reasons are:</p> <p>SK_INITIAL_CONFIG<br/>                     SK_DYNAMIC_CONFIG<br/>                     SK_MATRIX_STATE<br/>                     SK_RECEIPT_OF_NSR<br/>                     SK_CONFIG_TAG_RESET<br/>                     SK_CARD_RECONFIG<br/>                     SK_CARD_FAILURE</p> |
| NumMsgsSent   | This field indicates the number of messages sent to the CSP through SwitchManager.                                                                                                                                                                                                                                                                                                  |
| NumMsgsNacked | This field indicates the number of messages that have been negatively acknowledged by the CSP. If a config message gets a nack with status “No Ack From Switch” or “Card Not ready for configuration”, SwitchManager will try to re-send the config message several times before quitting. This would be reflected on the NumMsgsSent and NumMsgsNacked fields.                     |
| Slot          | <p>SK_CARD_FAILURE<br/>                     SK_CARD_RECONFIG<br/>                     SK_CONFIG_TAG_RESET</p>                                                                                                                                                                                                                                                                       |
| CardType      | <p>SK_CARD_FAILURE<br/>                     SK_CARD_RECONFIG<br/>                     SK_CONFIG_TAG_RESET</p>                                                                                                                                                                                                                                                                       |

**Relevant Attributes to ConfigType**

| ConfigType        | Attributes |
|-------------------|------------|
| SK_MATRIX_STATE   | Node       |
| SK_DYNAMIC_CONFIG | Nothing    |

## ConfigStatusMsg

|                     |                      |
|---------------------|----------------------|
| SK_RECEIPT_OF_NSR   | Node                 |
| SK_CARD_FAILURE     | Node, Slot, CardType |
| SK_CARD_RECONFIG    | Node, Slot, CardType |
| SK_CONFIG_TAG_RESET | Node, Slot, CardType |

# ConfigSwitch

---

**Type:** SwitchKit API message

**Purpose** Use the *SK\_ConfigSwitch* message to dynamically configure the switch without sending a configuration file.

**Description** *SK\_ConfigSwitch* uses the bytes of the message as parameters. The acknowledgement *SK\_ConfigSwitchAck* contains a status value from SwitchKit and a status value from the switch.

**How to use SK\_ConfigSwitch** You have to declare a C Structure to contain a configuration that will be sent to the switch. Use the function *sk\_packMessage*(MsgStruct \*m, char \*data, int \*NumBytes) to find out the fields NumBytes and Data. Send the message to the LLC using its member function send(). Wait for an acknowledgment, the status fields give information about the configuration message.

When you send the *SK\_ConfigSwitch* message, SwitchManager extracts it using the function *sk\_unpackMessage*() and sends the extracted information to the switch. Before SwitchManager can send an acknowledgment, it must wait until the switch processes the configuration. During this time, it is not possible to send another *SK\_ConfigSwitch* message to the switch. If another *SK\_ConfigSwitch* message is sent before receiving an acknowledgment, SwitchManager will NACK the second message. The SKStatus field will indicate this with SK\_NOT\_READY.

**Sent by** CSA or Application

**Field Values** The following table shows the possible field values for this message:

| Field   | Values and Description                                                                                                                                                                                                                                                   |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Action  | <ul style="list-style-type: none"> <li>0 = Do not send or process the message yet, there are more to come. SwitchManager stores the message in a TEMP collection.</li> <li>1 = Send this message and all the messages previously added using Action set to 0.</li> </ul> |
| Offline | <ul style="list-style-type: none"> <li>0 = Process and send the message.</li> <li>1 = Process the message but do not send it.</li> </ul>                                                                                                                                 |

**C Structure** typedef struct {  
 UBYTE Target;  
 UBYTE Type;  
 UBYTE MsgFormat;  
 UBYTE Action;  
 UBYTE Offline;  
 int Tag1;  
 int Tag2;  
 char AppGroupTarget[32];  
 int DataSize;  
 UBYTE Data[204];  
 } ***SK\_ConfigSwitch***;

**C Structure Response** typedef struct {  
 int Status;  
 int XLStatus;  
 } ***SK\_ConfigSwitchAck***;

**C++ Class** class ***SKC\_ConfigSwitch*** : public SKC\_ToolkitMessage {  
 public:  
 UBYTE getTarget() const;  
 void setTarget(UBYTE x);  
 UBYTE getType() const;  
 void setType(UBYTE x);  
 UBYTE getMsgFormat() const;  
 void setMsgFormat(UBYTE x);  
 UBYTE getAction() const;  
 void setAction(UBYTE x);  
 UBYTE getOffline() const;  
 void setOffline(UBYTE x);  
 int getTag1() const;  
 void setTag1(int x);  
 int getTag2() const;  
 void setTag2(int x);  
 const char \*getAppGroupTarget() const;  
 void setAppGroupTarget(const char \*x);  
 int getDataSize() const;  
 void setDataSize(int x);  
 const UBYTE \*getData() const;  
 UBYTE \*getData();  
 void setData(UBYTE \*x);  
 };

**C++ Class Response** class ***SKC\_ConfigSwitchAck*** : public SKC\_ToolkitAck {  
 public:  
 int getStatus() const;  
 void setStatus(int x);  
 int getXLStatus() const;  
 void setXLStatus(int x);  
 };

ConfigSwitch

```
};
```

## Conference Create 0x004B

---

**SwitchKit Name** ConferenceCreate

**Type** EXS API and SwitchKit API message

**Description** **Conference Create 0x004B**

**NOTE:** If you are using the DSP Series 2 card, use the *Resource Create 0x0124* message to create a conference.

Use this message to create a multi-channel conference on the MFDS or DSP-ONE card. A DSP configured for conferencing may have more than one conference created on it, in any combination of the maximum allowed channels.

Use the Conference ID returned by the CSP for any further references to the conference. The CSP assigns Conference IDs in descending order as they are created.

You receive a NACK of 0x2D (Incompatible PCM Encoding for Conference) if a channel from a remote node tries to connect 1-Way to a conference and the encoding type does not match the broadcast type set during the conference creation.

You cannot use the following Conference Types with the DSP Series 2 card:

- A-law Standard
- $\mu$ -law Standard
- Mixed

**Sent by** Host

**Example Message  
(Socket Log Output for  
SwitchKit)**

For EXS API, the following example message configures a unified conference of 25 channels with broadcast enabled.

For SwitchKit, the following socket log output shows configuration of a unified conference of 25 channels with broadcast enabled.

```
00 08 00 4B 00 00 FF 19 25 01
```

```
00 08 Message Length
```

```
00 4B Message Type
```

```
00 Reserved
```

```
00 Sequence Number
```

```
FF Logical Node ID (Single Node)
```

```
19 Conference Size (25 channels)
```

```
25 Conference Type (Any channel, Any DSP Unified)
```

```
01 Broadcast Enable (Broadcast Enabled)
```

The following example is for any channel, any DSP, Unified Dynamic, u-law conference of 25 channels, with broadcast enabled:

```
00 08 Message Length
00 4B Message Type
00 Reserved
00 Sequence Number
FF Logical Node ID (Single Node)
19 Conference Size (25 channels)
27 Conference Type
 (Any channel, Any DSP, Unified Dynamic, μ-law)
01 Broadcast Enable (Broadcast Enabled)
```

### SwitchKit Code C Structure

```
typedef struct {
 UBYTE ConferenceSize;
 UBYTE ConferenceType;
 UBYTE BroadcastEnable;
} XL_ConferenceCreate;
```

### C Structure Response

```
typedef struct {
 unsigned short Status;
 unsigned short ConferenceID;
} XL_ConferenceCreateAck;
```

### C++ Class

```
class XLC_ConferenceCreate : public XLC_OutboundMessage {
public:
 UBYTE getConferenceSize() const;
 void setConferenceSize(UBYTE x);
 UBYTE getConferenceType() const;
 void setConferenceType(UBYTE x);
 UBYTE getBroadcastEnable() const;
 void setBroadcastEnable(UBYTE x);
};
```

### C++ Class Response

```
class XLC_ConferenceCreateAck : public
 XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 unsigned short getConferenceID() const;
 void setConferenceID(unsigned short x);
};
```

### EXS API Hex Format

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | RESPONSE (Gray) |                       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|----------------|--------------------|-------|--------------------|---------|--------------------|------------|-----------------------|--|-----------------------------|--------------|-----------------------|--|-----------------------------|------|-------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Byte            | Field Description     |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 0               | Frame (0xFE)          |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| 1, 2            | Length (0x0008)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 1, 2            | Length (0x0009)       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| 3, 4            | Message Type (0x004B)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 3, 4            | Message Type (0x004B) |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 5               | Reserved (0x00)       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| 6               | Sequence Number                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 6               | Same Sequence Number  |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 7               | Logical Node ID       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| 8               | <p>Conference Size</p> <p>The number of channels to be included in the conference. The allowed size range for each conference type is shown below.</p> <table border="0"> <tr> <td>A-law Standard</td> <td>2-7 (DSP-ONE only)</td> </tr> <tr> <td>μ-law Standard</td> <td>2-7 (DSP-ONE only)</td> </tr> <tr> <td>Mixed</td> <td>2-9 (DSP-ONE only)</td> </tr> <tr> <td>Monitor</td> <td>2-9 (DSP-ONE only)</td> </tr> <tr> <td>Unified</td> <td>2-25 for DSP-ONE card</td> </tr> <tr> <td></td> <td>2-128 for DSP Series 2 card</td> </tr> <tr> <td>DTMF Clamped</td> <td>2-24 for DSP-ONE card</td> </tr> <tr> <td></td> <td>2-128 for DSP Series 2 card</td> </tr> </table> | A-law Standard  | 2-7 (DSP-ONE only)    | μ-law Standard | 2-7 (DSP-ONE only) | Mixed | 2-9 (DSP-ONE only) | Monitor | 2-9 (DSP-ONE only) | Unified    | 2-25 for DSP-ONE card |  | 2-128 for DSP Series 2 card | DTMF Clamped | 2-24 for DSP-ONE card |  | 2-128 for DSP Series 2 card | 8, 9 | Status (MSB, LSB) |
|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | A-law Standard  | 2-7 (DSP-ONE only)    |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | μ-law Standard  | 2-7 (DSP-ONE only)    |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| Mixed           | 2-9 (DSP-ONE only)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                 |                       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| Monitor         | 2-9 (DSP-ONE only)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                 |                       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| Unified         | 2-25 for DSP-ONE card                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                 |                       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
|                 | 2-128 for DSP Series 2 card                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                 |                       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| DTMF Clamped    | 2-24 for DSP-ONE card                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                 |                       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
|                 | 2-128 for DSP Series 2 card                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                 |                       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| 10, 11          | <p>Conference ID (MSB, LSB)</p> <p>The Conference ID returned by the CSP is 16 bits long. The host uses this conference ID for any future references to this conference. For a local conference, all the node ID bits are set to 1.</p> <table border="0"> <tr> <td>Bits 0-8</td> <td>Local Conference ID</td> </tr> <tr> <td>Bit 9</td> <td>Monitor Bit</td> </tr> <tr> <td></td> <td>0 Non-Monitor</td> </tr> <tr> <td></td> <td>1 Monitor</td> </tr> <tr> <td>Bits 10-15</td> <td>Node ID</td> </tr> </table>                                                                                                                                                              | Bits 0-8        | Local Conference ID   | Bit 9          | Monitor Bit        |       | 0 Non-Monitor      |         | 1 Monitor          | Bits 10-15 | Node ID               |  |                             |              |                       |  |                             |      |                   |
| Bits 0-8        | Local Conference ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                 |                       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| Bit 9           | Monitor Bit                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                 |                       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
|                 | 0 Non-Monitor                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                 |                       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
|                 | 1 Monitor                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                 |                       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| Bits 10-15      | Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                 |                       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| 12              | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                 |                       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |
| 9               | <p>Conference Type</p> <p>The value of this field is divided into nibbles. The upper nibble represents channel and DSP allocation within the conference. The lower nibble represents the conference type.</p> <p>This format is backward-compatible with legacy versions of system software. If you enter 0 in the upper nibble, your existing applications are not affected.</p>                                                                                                                                                                                                                                                                                             |                 |                       |                |                    |       |                    |         |                    |            |                       |  |                             |              |                       |  |                             |      |                   |

Conference Create 0x004B

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    | <p>Channel/DSP Pool (upper nibble):</p> <ul style="list-style-type: none"> <li>0x0N Local Channels/Local DSP</li> <li>0x1N Any Channels/Local DSP. Cannot be used without EXS Conferencing over the EXNET® ring configured.</li> <li>0x2N Any Channels/Any DSP. Cannot be used without EXS Conferencing over the EXNET® ring configured.</li> </ul> <p>The Any Channel option in the Channel/DSP Pool conference type is available on EXNET® systems (multi-node switches) only, after the <i>EXS Node Configure 0x007F</i> message (Entity 0x0001 Number of Timeslots) has been sent.</p> <p>Conference Type (lower nibble):</p> <ul style="list-style-type: none"> <li>0xN1 <math>\mu</math>-law Standard</li> <li>0xN2 A-law Standard</li> <li>0xN3 Mixed (<math>\mu</math>-law and A-law encoded)</li> <li>0xN4 Monitor</li> <li>0xN5 Unified</li> <li>0xN6 DTMF-Clamped</li> <li>0xN7 Unified Dynamic, <math>\mu</math>-law Broadcast</li> <li>0xN8 Unified Dynamic, <math>\mu</math>-law Broadcast with DTMF Clamping Support</li> <li>0xN9 Unified Dynamic, A-law Broadcast</li> <li>0xNA Unified Dynamic, A-law Broadcast with DTMF Clamping Support</li> </ul> |
| 10 | <p>Broadcast Enable</p> <p><b>NOTE:</b> Broadcasting is not supported for mixed conferences.</p> <ul style="list-style-type: none"> <li>0x00 Broadcast Not Enabled<br/>Mixed conferences should be set to 0, because broadcasting is not supported.</li> <li>0x01 Broadcast Enabled<br/>Full conference output is always provided for A-law, u-law, Unified Dynamic Broadcast, and Monitor conferences. The full output can be broadcast to a channel using the <i>Connect One-Way to Conference</i> message.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 11 | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

# Conference Delete Request 0x004C

---

|                       |                                         |
|-----------------------|-----------------------------------------|
| <b>SwitchKit Name</b> | ConferenceDeleteRequest                 |
| <b>Type</b>           | EXS API and SwitchKit API message       |
| <b>Description</b>    | <b>Conference Delete Request 0x004C</b> |

**NOTE:** If you are using the DSP Series 2 card, use the *Resource Delete 0x0126* message to delete a conference.

Use this message to delete a conference on the MFDSP or DSP-ONE card. A response *Status* value of positive acknowledgment (0x0010) indicates that conference deletion has been “initiated” by the CSP. When deletion is complete, the CSP will send a *Conference Deleted* (0x004D) message to the host.

To delete all conferences, specify **0xFFFF** in the *Conference ID* field.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short ConferenceID;
 UBYTE ForcedFlag;
} XL_ConferenceDeleteRequest;
```

**C Structure Response**

```
typedef struct {
 unsigned short Status;
 unsigned short ConferenceID;
} XL_ConferenceDeleteRequestAck;
```

**C++ Class**

```
class XL_ConferenceDeleteRequest : public
 XLC_OutboundMessage {
public:
 unsigned short getConferenceID() const;
 void setConferenceID(unsigned short x);
 UBYTE getForcedFlag() const;
 void setForcedFlag(UBYTE x);
};
```

**C++ Class Response**

```
class XL_ConferenceDeleteRequestAck : public
 XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 unsigned short getConferenceID() const;
 void setConferenceID(unsigned short x);
```

};

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                                   | RESPONSE (Gray) |                          |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                                                 | Byte            | Field Description        |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                      | 0               | Frame (0xFE)             |
| 1, 2            | Length, (0x00NN)                                                                                                                                                                                                                                                                                                  | 1, 2            | Length (0x0009)          |
| 3, 4            | Message Type (0x004C)                                                                                                                                                                                                                                                                                             | 3, 4            | Message Type (0x004C)    |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                   | 5               | Reserved (0x00)          |
| 6               | Sequence Number                                                                                                                                                                                                                                                                                                   | 6               | Same Sequence Number     |
| 7               | Logical Node ID                                                                                                                                                                                                                                                                                                   | 7               | Logical Node ID          |
|                 |                                                                                                                                                                                                                                                                                                                   | 8, 9            | Status (MSB, LSB)        |
|                 |                                                                                                                                                                                                                                                                                                                   | 10, 11          | Conference ID (MSB, LSB) |
|                 |                                                                                                                                                                                                                                                                                                                   | 12              | Checksum                 |
| 8, 9            | Conference ID (MSB, LSB)<br>Number assigned to the conference in the Response to the Conference Create message.<br><b>NOTE:</b> For this field, the value 0xFFFF has a special meaning (Delete all conferences)                                                                                                   |                 |                          |
| 10              | Forced Flag<br>0x00 Graceful deletion<br>The conference will be deleted after all the channels in the conference are released.<br><br>0x01 Forced deletion<br>All channels involved in the conference will be released or parked (according to their configured Release Mode) and the conference will be deleted. |                 |                          |
| 11              | Checksum                                                                                                                                                                                                                                                                                                          |                 |                          |

# Conference Deleted 0x004D

---

**SwitchKit Name** ConferenceDeleted

**Type** EXS API and SwitchKit API message

**Description** **Conference Deleted 0x004D**

**NOTE:** If you are using the DSP Series 2 card, you will receive a *Resource Delete Indication 0x0129* message to indicate a conference has been deleted.

This message is sent to the host after a conference on an MFDSP or DSP-ONE card is deleted (for example, all channels in the conference are released and the DSP resource is freed up).

**Sent by** CSP

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short ConferenceID;
} XL_ConferenceDeleted;
```

**C++ Class**

```
class XLC_ConferenceDeleted : public XLC_InboundMessage {
public:
 unsigned short getConferenceID() const;
 void setConferenceID(unsigned short x);
};
```

**Resent in EXS API** This message is resent once after five seconds.

## EXS API Hex Format

| MESSAGE (White) |                       | RESPONSE (Gray) |                       |
|-----------------|-----------------------|-----------------|-----------------------|
| Byte            | Field Description     | Byte            | Field Description     |
| 0               | Frame (0xFE)          | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x0007)       | 1, 2            | Length (0x0005)       |
| 3, 4            | Message Type (0x004D) | 3, 4            | Message Type (0x004D) |
| 5               | Reserved (0x00)       | 5               | Reserved (0x00)       |
| 6               | Sequence Number       | 6               | Same Sequence Number  |
| 7               | Logical Node ID       | 7               | Logical Node ID       |
| 8, 9            | Conference ID         | 8               | Checksum              |
| 10              | Checksum              |                 |                       |

## Connect 0x0000

---

**SwitchKit Name** Connect

**Type** EXS API and SwitchKit API message

**Description** **Connect 0x0000**

This message instructs the CSP to establish a two-way connection between the two channels specified, for example, Channel A and Channel B.

If there is no active call up for Channel B, a seizure will be generated on that channel, and once seize ACKs have been met, the voice path is connected.

**NOTE:** This message treats the first span/channel and the second span/channel the same regardless of their inbound or outbound nature. Therefore, the answer supervision mode of only the second span/channel is meaningful in the *Connect* API message.

**Sent by** Host

**Example Message (Socket Log Output for SwitchKit)**

For EXS API, the example message below configures the CSP to establish a two-way connection between span 01 channel 01 and span 02 channel 02:  
For SwitchKit, the socket log output below shows configuration of the CSP, which establishes a two-way connection between span 01 channel 01 and span 02 channel 02:

```
00 11 00 00 00 00 FF 00 02 0D 03 00 01 01 0D 03 00 02 02
```

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short SpanA;
 UBYTE ChannelA;
 unsigned short SpanB;
 UBYTE ChannelB;
} XL_Connect;
```

**C++ Class**

```
class XLC_Connect : public XLC_OutboundMessage {
public:
 unsigned short getSpanA() const;
 void setSpanA(unsigned short x);
 UBYTE getChannelA() const;
 void setChannelA(UBYTE x);
 unsigned short getSpanB() const;
 void setSpanB(unsigned short x);
 UBYTE getChannelB() const;
 void setChannelB(UBYTE x);
```

};

**EXS API Hex Format**

| MESSAGE (White)                         |                                     | RESPONSE (Gray) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------------------|-------------------------------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Byte                                    | Field Description                   | Byte            | Field Description                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 0                                       | Frame (0xFE)                        | 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 1, 2                                    | Length (0x0011)                     | 1, 2            | Length (0x0007)                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 3, 4                                    | Message Type (0x0000)               | 3, 4            | Message Type (0x0000)                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 5                                       | Reserved (0x00)                     | 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 6                                       | Sequence Number                     | 6               | Same Sequence Number                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 7                                       | Logical Node ID                     | 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| :                                       | AIB                                 | 8, 9            | Status (MSB, LSB)<br>See Response Status Values.                                                                                                                                                                                                                                                                                                                                                                                                                               |
|                                         | Address Method<br>0x00 - Individual | 10,11           | State (MSB, LSB)<br>If the MSB of the preceding Status field is 0x18 (Invalid Channel B State) or 0x1D (Invalid Channel A State), then this field indicates the component state. If the State value is greater than 0xFF (255), then it is truncated to 0xFF.<br><br>If the MSB of the preceding Status field is neither 0x18 nor 0x1D, then the value in this field is 0x0000.<br><br>If the preceding Status field is a positive acknowledgement, this field does not apply. |
|                                         | Number of AEs to follow<br>0x02     |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| AEs<br>0x0D Channel A<br>0x0D Channel B |                                     |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| :                                       | Checksum                            | 12              | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

## Connect One-Way To Conference 0x004F

---

**SwitchKit Name** ConnectOneWayToConference

**Type** EXS API and SwitchKit API message

**Description** **Connect One-Way To Conference 0x004F**

**NOTE:** If you are using the DSP Series 2 card, use the *Resource Connect 0x0127* message to connect to a conference. Use the Connection Type TLV to make a one-way connection.

Use this message with the MFDSP or DSP-ONE card to establish a one-way (listen only) connection between the channel specified and the previously created conference specified. The channel should be in one of the following states: idle, in seized, parked, or out seized.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
 unsigned short ConferenceID;
} XL_ConnectOneWayToConference;
```

### C Structure Response

```
typedef struct {
 unsigned short Status;
 unsigned short Span;
 UBYTE Channel;
} XL_ConnectOneWayToConferenceAck;
```

### C++ Class

```
class XLC_ConnectOneWayToConference : public
 XLC_OneChannelOutbound {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 unsigned short getConferenceID() const;
 void setConferenceID(unsigned short x);
};
```

### C++ Class Response

```
class XLC_ConnectOneWayToConferenceAck : public
 XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
```

Connect One-Way To Conference 0x004F

```

unsigned short getSpan() const;
void setSpan(unsigned short x);
UBYTE getChannel() const;
void setChannel(UBYTE x);
};

```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                            | RESPONSE (Gray) |                                                                                                                                                                  |
|-----------------|----------------------------------------------------------------------------------------------------------------------------|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Byte            | Field Description                                                                                                          | Byte            | Field Description                                                                                                                                                |
| 0               | Frame (0xFE)                                                                                                               | 0               | Frame (0xFE)                                                                                                                                                     |
| 1, 2            | Length (0x00NN)                                                                                                            | 1, 2            | Length (0x00NN)                                                                                                                                                  |
| 3, 4            | Message Type (0x004F)                                                                                                      | 3, 4            | Message Type (0x004F)                                                                                                                                            |
| 5               | Reserved (0x00)                                                                                                            | 5               | Reserved (0x00)                                                                                                                                                  |
| 6               | Sequence Number                                                                                                            | 6               | Same Sequence Number                                                                                                                                             |
| 7               | Logical Node ID                                                                                                            | 7               | Logical Node ID                                                                                                                                                  |
| :               | AIB<br>Address Method<br>0x00 - Individual AEs                                                                             | 8, 9            | Status (MSB, LSB)<br>0x1D03 Invalid Channel State: Channel Idle<br>0x1D09 Invalid Channel State: L3 CLear Wait<br>Also see Common Response Status Values chapter |
|                 | Number of AEs to follow                                                                                                    |                 |                                                                                                                                                                  |
|                 | AEs<br>0x0D Channel                                                                                                        |                 |                                                                                                                                                                  |
| :               | Conference ID (MSB, LSB)<br>The number assigned to the conference in the Response to the <i>Conference Create</i> message. | 1:              | AIB<br>Same as message                                                                                                                                           |
| :               | Checksum                                                                                                                   | :               | Checksum                                                                                                                                                         |

## Connect To Conference 0x004E

---

**SwitchKit Name** ConnectToConference

**Type** EXS API and SwitchKit API message

**Description** **Connect To Conference 0x004E**

**NOTE:** If you are using the DSP Series 2 card, use the *Resource Connect 0x0127* message to connect to a conference.

Use this message to add a channel to a conference on an MFDSP or DSP-ONE card. The channel to be added to a conference must be in one of the following states: inseized, parked, connected, or outseized.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
 unsigned short ConferenceID;
} XL_ConnectToConference;
```

### C Structure Response

```
typedef struct {
 unsigned short Status;
 unsigned short Span;
 UBYTE Channel;
} XL_ConnectToConferenceAck;
```

### C++ Class

```
class XLC_ConnectToConferenceAck : public
 XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
};
```

### C++ Class Response

```
class XLC_ConnectToConference : public
 XLC_OneChannelOutbound {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 unsigned short getConferenceID() const;
```

```
void setConferenceID(unsigned short x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                            | RESPONSE (Gray) |                                                                                                                                                                  |
|-----------------|----------------------------------------------------------------------------------------------------------------------------|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Byte            | Field Description                                                                                                          | Byte            | Field Description                                                                                                                                                |
| 0               | Frame (0xFE)                                                                                                               | 0               | Frame (0xFE)                                                                                                                                                     |
| 1, 2            | Length (0x00NN)                                                                                                            | 1, 2            | Length (0x00NN)                                                                                                                                                  |
| 3, 4            | Message Type (0x004E)                                                                                                      | 3, 4            | Message Type (0x004E)                                                                                                                                            |
| 5               | Reserved (0x00)                                                                                                            | 5               | Reserved (0x00)                                                                                                                                                  |
| 6               | Sequence Number                                                                                                            | 6               | Same Sequence Number                                                                                                                                             |
| 7               | Logical Node ID                                                                                                            | 7               | Logical Node ID                                                                                                                                                  |
| :               | <u>AIB</u><br>0x00 - Individual AEs<br>Number of AEs to follow<br>AE<br>0x0D Channel                                       | 8, 9            | Status (MSB, LSB)<br>0x1D03 Invalid Channel State: Channel Idle<br>0x1D09 Invalid Channel State: L3 CLear Wait<br>Also see Common Response Status Values chapter |
| :               | Conference ID (MSB, LSB)<br>The number assigned to the conference in the Response to the <i>Conference Create</i> message. | :               | <u>AIB</u><br>Same as message                                                                                                                                    |
| :               | Checksum                                                                                                                   | :               | Checksum                                                                                                                                                         |

# Connect One-Way Forced 0x0050

---

**SwitchKit Name** ConnectOneWayForced

**Type** EXS API and SwitchKit API message

**Description** **Connect One-Way Forced 0x0050**

This message establishes a one-way connection between two channels, where Channel A is the source channel (talk only) and Channel B is the destination channel (listen only).

With this connection, the CSP does not do any checking on the state of Channel A; it can be in any state. This lessens the degree of call processing required, resulting in a significantly faster call rate. Channel B, however, must be in an acceptable connection state.

The connection will only be torn down if Channel B is released by the host, or as a result of a network-initiated release.

When performing a “Connect One-Way Forced A-to-B” on a multi-node system, you must send this message to the node containing the “B” channel.

For SwitchKit: If you use the ConnectOneWayForced message in your call control application, you need to specify which node you want the message to be sent to. You can specify the node within the C Structure field BaseFields Base (NodeID) or within in the C++ Class through inheritance from SKC\_Message (getNode, setNode).

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short SpanA;
 UBYTE ChannelA;
 unsigned short SpanB;
 UBYTE ChannelB;
 UBYTE BPad;
} XL_ConnectOneWayForced;
```

**C++ Class**

```
class XLC_ConnectOneWayForced : public
 XLC_OutboundMessage {
public:
 unsigned short getSpanA() const;
 void setSpanA(unsigned short x);
 UBYTE getChannelA() const;
 void setChannelA(UBYTE x);
 unsigned short getSpanB() const;
 void setSpanB(unsigned short x);
 UBYTE getChannelB() const;
 void setChannelB(UBYTE x);
 UBYTE getBPad() const;
 void setBPad(UBYTE x);
```

};

**EXS API Hex Format**

| MESSAGE (White)                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | RESPONSE (Gray) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|--------------|------|-------|------|------|------|------|------|------|------|------|------|------|------|------|
| Byte                                    | Field Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Byte            | Field Description                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| 0                                       | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| 1, 2                                    | Length (0x00NN)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 1, 2            | Length (0x0007)                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| 3, 4                                    | Message Type (0x0050)                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 3, 4            | Message Type (0x0050)                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| 5                                       | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| 6                                       | Sequence Number                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 6               | Same Sequence Number                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| 7                                       | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| :                                       | AIB<br>Address Method<br>0x00 - Individual AEs                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 8, 9            | Status (MSB, LSB)                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
|                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 10, 11          | State (MSB, LSB)<br>If the MSB of the preceding Status field is 0x18 (Invalid Channel B State) or 0x1D (Invalid Channel A State), then this field indicates the component state. If the State value is greater than 0xFF (255), then it is truncated to 0xFF.<br><br>If the MSB of the preceding Status field is neither 0x18 nor 0x1D, then the value in this field is 0x0000.<br><br>If the preceding Status field is a positive acknowledgement, this field does not apply. |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
|                                         | Number of AEs to follow                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 12              | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| AEs<br>0x0D Channel A<br>0x0D Channel B |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| :                                       | Channel B Pad Value<br>The db gain/loss adjustment of the signal being transmitted to Channel B.<br><table border="1"> <thead> <tr> <th>Value</th> <th>dB Gain/Loss</th> </tr> </thead> <tbody> <tr> <td>0x00</td> <td>+3 dB</td> </tr> <tr> <td>0x01</td> <td>0 dB</td> </tr> <tr> <td>0x02</td> <td>2 dB</td> </tr> <tr> <td>0x03</td> <td>3 dB</td> </tr> <tr> <td>0x04</td> <td>4 dB</td> </tr> <tr> <td>0x05</td> <td>6 dB</td> </tr> <tr> <td>0x06</td> <td>9 dB</td> </tr> </tbody> </table> |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Value | dB Gain/Loss | 0x00 | +3 dB | 0x01 | 0 dB | 0x02 | 2 dB | 0x03 | 3 dB | 0x04 | 4 dB | 0x05 | 6 dB | 0x06 | 9 dB |
| Value                                   | dB Gain/Loss                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| 0x00                                    | +3 dB                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| 0x01                                    | 0 dB                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| 0x02                                    | 2 dB                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| 0x03                                    | 3 dB                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| 0x04                                    | 4 dB                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| 0x05                                    | 6 dB                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| 0x06                                    | 9 dB                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
|                                         | Number of ICBs to follow                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
|                                         | Data (presented in an ICB)<br><br>0x03 Extended Data<br><br>0x00E1 Generic PPL                                                                                                                                                                                                                                                                                                                                                                                                                      |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |
| :                                       | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |              |      |       |      |      |      |      |      |      |      |      |      |      |      |      |



## Connect Tone Pattern 0x002F

---

**SwitchKit Name** ConnectTonePattern

**Type** EXS API and SwitchKit API message

**Description** **Connect Tone Pattern 0x002F**

This message instructs the CSP to transmit a call progress pattern ID to the specified channel. The channel must be in any state other than: idle, out of service, or busied out.

If you want to be notified with a *Call Processing Event* message (0x21 - Tone Complete) when tone transmission has stopped, set the *Generate Event Flag*.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
 UBYTE reserved20[27];
 UBYTE CallProgressPatternID;
 unsigned short NumCycles;
 UBYTE Flag;
 UBYTE reserved51[219];
} XL_ConnectTonePattern;
```

**C++ Class**

```
class XLC_ConnectTonePattern : public
 XLC_OneChannelOutbound {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 UBYTE getCallProgressPatternID() const;
 void setCallProgressPatternID(UBYTE x);
 unsigned short getNumCycles() const;
 void setNumCycles(unsigned short x);
 UBYTE getFlag() const;
 void setFlag(UBYTE x)
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | RESPONSE (Gray) |                       |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x002F)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 3, 4            | Message Type (0x002F) |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 7               | Logical Node ID       |
| 8               | AIB (Individual AEs)<br>0x0D Channel                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 8, 9            | Status (MSB, LSB)     |
|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 10              | Checksum              |
| :               | Transmit Call Progress Pattern ID<br><b>NOTE:</b> These are <b>not</b> the same pattern IDs as those used for call progress analysis.<br><br>0x00 Silence Tone<br>0x01 Dial Tone<br>0x02 Ringback<br>0x03 Busy<br>0x04 Reorder<br>0x05 Warning<br>0x06 Call Waiting<br>0x07 ONI Call (Zip Tone)<br>0x08 ANI Failure (Zip Tone)<br>0x09 Confirmation Tone<br>0x0A Recall Dial Tone<br>0x0B Class of Service, Tone 2<br>0x0C Class of Service, Tone 3<br>0x0D Intercept<br>0x0E Vacant Code<br>0x0F Reorder-LEC<br>0x10 No Circuit-LEC<br>0x11 Reorder-Carrier<br>0x12 No Circuit-Carrier<br>0x13 400 Hz continuous tone<br>0x14 Specialized tone to wait 500 ms then play 480Hz for 800 ms<br>0x15 Bong Tone (Calling Card Service Prompt Tone) (When specifying the Bong Tone, it will only play once, but not a number of cycles).<br>0x16-3F User-defined Patterns |                 |                       |

Connect Tone Pattern 0x002F

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | <p>Number of Cycles to Transmit, MSB, LSB</p> <p>The value in this field for ringback should be greater than 0x03 and for all others greater than 0x01.</p> <p>The value in this field does not apply for the Dial Tone (0x01), Recall Dial Tone (0x0A), and Bong Tone (0x15), as they have no cycles. Dial Tone and Recall Dial Tone play continuously and Bong Tone plays once.</p> <p>0x0000 - undefined<br/>         0x0000–0xFFFE Valid numbers of cycles<br/>         0xFFFF Continuous (for example, Dial Tone)</p>                                                                                                                                                                                                                                                                                                                                                              |
| : | <p>Generate Event Flag</p> <p>0x00 Do not inform the host when the outputting of the tone is complete.<br/>         0x01 Inform the host with a <i>Call Processing Event</i> message that the outputting of the tone is complete.</p> <p>The CSP stops playing the tone only if:</p> <ul style="list-style-type: none"> <li>- the number of cycles to transmit is reached</li> <li>- the host issues a <i>Collect Digit String</i> message with Configuration Bit 4 set to Cancel Prompting Tone or RAN (0x00)</li> <li>- the host issues a <i>Disconnect Tone Pattern</i> message.</li> </ul> <p>A <i>Connect Tone Pattern</i> message does not cancel the outputting of call progress tones. However, a <i>Connect With Data</i> message with an action ICB of Free System Resources does cancel outputting. For more information, refer to the <i>Connect With Data</i> message.</p> |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

# Connect Wait 0x0017

---

**SwitchKit Name** ConnectWait

**Type** EXS API and SwitchKit API message

**Description** **Connect Wait 0x0017**

This message instructs the CSP to suspend the processing of an incoming call on a channel. This message may be used by the host as a response to the *Request for Service* message sent by the CSP.

The CSP stops resending the *Request for Service* message for the channel and waits for a connection management message such as *Park Channel*, *Connect*, and so on.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
} XL_ConnectWait;
```

**C Structure Response**

```
typedef struct {
 unsigned short Status;
 unsigned short Span;
 UBYTE Channel;
} XL_ConnectWaitAck;
```

**C++ Class**

```
class XLC_ConnectWait : public XLC_OneChannelOutbound {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
};
```

**C++ Class Response**

```
class XLC_ConnectWaitAck : public XLC_AcknowledgeMessage
{
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                | RESPONSE (Gray) |                                                                                                                                                                  |
|-----------------|------------------------------------------------|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Byte            | Field Description                              | Byte            | Field Description                                                                                                                                                |
| 0               | Frame (0xFE)                                   | 0               | Frame (0xFE)                                                                                                                                                     |
| 1, 2            | Length (0x00NN)                                | 1, 2            | Length (0x00NN)                                                                                                                                                  |
| 3, 4            | Message Type (0x0017)                          | 3, 4            | Message Type (0x0017)                                                                                                                                            |
| 5               | Reserved (0x00)                                | 5               | Reserved (0x00)                                                                                                                                                  |
| 6               | Sequence Number                                | 6               | Same Sequence Number                                                                                                                                             |
| 7               | Logical Node ID                                | 7               | Logical Node ID                                                                                                                                                  |
| :               | AIB<br>Address Method<br>0x00 - Individual AEs | 8, 9            | Status (MSB, LSB)<br>0x1D03 Invalid Channel State: Channel Idle<br>0x1D09 Invalid Channel State: L3 CLeAr Wait<br>Also see Common Response Status Values chapter |
|                 | Number of AEs to follow                        |                 |                                                                                                                                                                  |
|                 | AEs<br>0x0D Channel                            |                 |                                                                                                                                                                  |
| :               | Checksum                                       | :               | AIB<br>Same as message                                                                                                                                           |
|                 |                                                | :               | Checksum                                                                                                                                                         |

## Connect With Data 0x0005

---

|                                    |                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b>              | ConnectWithData                                                                                                                                                                                                                                                                                                                                                                     |
| <b>Type</b>                        | EXS API and SwitchKit API message                                                                                                                                                                                                                                                                                                                                                   |
| <b>Description</b>                 | <p><b>Connect With Data 0x0005</b></p> <p>This message instructs the CSP to connect two channels.</p> <p>This message also supports canceling digit collection, playing of call progress tones, voice path connect functionality, and dynamic configuration of release modes.</p>                                                                                                   |
| <b>ISDN</b>                        | If the terminating channel is not in an idle state, after seizure the alerting and connect indications will get propagated to the incoming side and the data specified in this message will get sent to the incoming ISDN channel.                                                                                                                                                  |
| <b>SS7</b>                         | You can use this message to pass parameter data for the ACM, ANM, or CON (ITU-TS only) in a connection between an incoming SS7 call (A Party) and another channel.                                                                                                                                                                                                                  |
| <b>SIP (Call Agent)</b>            | The message carries the B side SDP (Session Description Protocol) information to the CSP in an NPDI Universal ICB and Generic PPL ICB. In a Generic PPL ICB, the Channel Service TLV indicates whether a bearer-switched or bearer-free connection is requested. For a bearer-free connection, the B side connection information is passed in the NPDI Universal ICB from the host. |
| <b>Sent by</b>                     | Host                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Example Message for EXS API</b> | <p>The following example message uses the Release Mode Configure ICB in a <i>Connect With Data</i> message to park Channel A (Span 0, Channel 0) and release Channel B (Span 0, Channel 1).</p> <pre>00 0D 00 05 00 00 FF 01 02 0D 03 00 00 00 0D 03 00 00 01   01 01 01 02 02 01 02</pre>                                                                                          |
| <b>SwitchKit Code</b>              | <p><b>C Structure</b></p> <pre>typedef struct {     unsigned short SpanA;     UBYTE ChannelA;     unsigned short SpanB;     UBYTE ChannelB;     UBYTE ConnectDataType;     UBYTE Data[222]; } <b>XL_ConnectWithData</b>;</pre> <p><b>C++ Class</b></p> <pre>class <b>XLC_ConnectWithData</b> : public XLC_OutboundMessage { public:</pre>                                           |

```

unsigned short getSpanA() const;
void setSpanA(unsigned short x);
UBYTE getChannelA() const;
void setChannelA(UBYTE x);
unsigned short getSpanB() const;
void setSpanB(unsigned short x);
UBYTE getChannelB() const;
void setChannelB(UBYTE x);
UBYTE getConnectDataType() const;
void setConnectDataType(UBYTE x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};

```

**EXS API Hex Format**

| MESSAGE (White) |                          | RESPONSE (Gray) |                       |
|-----------------|--------------------------|-----------------|-----------------------|
| Byte            | Field Description        | Byte            | Field Description     |
| 0               | Frame (0xFE)             | 0               | Frame (0xFE)          |
| 1, 2            | Length (MSB, LSB) 0x00NN | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x0005)    | 3, 4            | Message Type (0x0005) |
| 5               | Reserved (0x00)          | 5               | Reserved (0x00)       |
| 6               | Sequence Number          | 6               | Same Sequence Number  |
| 7               | Logical Node ID          | 7               | Logical Node ID       |

|                                         |                                                                                                                                                                                                                                                                                                                                                                                            |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| :                                       | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs                                                                                                                                                                                                                                                                                                                                      | 8, 9   | Status (MSB, LSB)<br><br>0x0001 Invalid TLV Data<br>Software cannot find the TLV Data Buffer. This can also occur if the Data for a TLV is out of range.<br><br>0x0004 Invalid TLV Length<br>The TLV length is different from what is expected.<br><br>0x0006 Invalid TLV<br>Unknown TLV<br><br>Also see Common Response Status Values in the <i>API Reference</i> .                                                                                                           |
|                                         |                                                                                                                                                                                                                                                                                                                                                                                            | 10, 11 | State (MSB, LSB)<br>If the MSB of the preceding Status field is 0x18 (Invalid Channel B State) or 0x1D (Invalid Channel A State), then this field indicates the component state. If the State value is greater than 0xFF (255), then it is truncated to 0xFF.<br><br>If the MSB of the preceding Status field is neither 0x18 nor 0x1D, then the value in this field is 0x0000.<br><br>If the preceding Status field is a positive acknowledgement, this field does not apply. |
|                                         |                                                                                                                                                                                                                                                                                                                                                                                            | 12     | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Number of AEs to follow<br>0x02         |                                                                                                                                                                                                                                                                                                                                                                                            |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| AEs<br>0x0D Channel A<br>0x0D Channel B |                                                                                                                                                                                                                                                                                                                                                                                            |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| :                                       | Connect Data Type<br>0x01 Use ICBs<br>0x02 Reserved                                                                                                                                                                                                                                                                                                                                        |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| :                                       | Number of ICBs to follow (Ignore this field if no ICBs in message.)                                                                                                                                                                                                                                                                                                                        |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| :                                       | Data (presented in an ICB)<br>0x01 Action ICBs<br>0x01 Free System Resources<br>0x02 Release Mode Configure<br>0x02 Data ICBs<br>0x12 SS7 Parameters<br>0x17 ISDN Formatted IEs With Event<br>0x18 ISDN Raw IEs with Event<br>0x1C SS7 TUP Formatted Fields<br>0x26 Channel Pad Value<br>0x03 Extended Data<br>0x001E Generic PPL<br>0x0026 Channel Pad Value<br>0x0033 NPDI Universal ICB |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| :                                       | Checksum                                                                                                                                                                                                                                                                                                                                                                                   |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

## Connect With Pad 0x0003

---

|                       |                                                                                                                                                                                                                           |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | ConnectWithPad                                                                                                                                                                                                            |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                                                                                                                         |
| <b>Description</b>    | <b>Connect With Pad 0x0003</b><br>This message instructs the CSP to establish a connection between the two channels specified while maintaining the dB gain/loss adjustment specified for the duration of the connection. |
| <b>Sent by</b>        | Host                                                                                                                                                                                                                      |

**SwitchKit Code**    **C Structure**

```
typedef struct {
 unsigned short SpanA;
 UBYTE ChannelA;
 unsigned short SpanB;
 UBYTE ChannelB;
 UBYTE APad;
 UBYTE BPad;
} XL_ConnectWithPad;
```

**C++ Class**

```
class XLC_ConnectWithPad : public XLC_OutboundMessage {
public:
 unsigned short getSpanA() const;
 void setSpanA(unsigned short x);
 UBYTE getChannelA() const;
 void setChannelA(UBYTE x);
 unsigned short getSpanB() const;
 void setSpanB(unsigned short x);
 UBYTE getChannelB() const;
 void setChannelB(UBYTE x);
 UBYTE getAPad() const;
 void setAPad(UBYTE x);
 UBYTE getBPad() const;
 void setBPad(UBYTE x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                   | RESPONSE (Gray) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------|---------------------------------------------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Byte            | Field Description                                 | Byte            | Field Description                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 0               | Frame (0xFE)                                      | 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 1, 2            | Length (0x00nn)                                   | 1, 2            | Length (0x0007)                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 3, 4            | Message Type (0x0003)                             | 3, 4            | Message Type (0x0003)                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 5               | Reserved (0x00)                                   | 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 6               | Sequence Number                                   | 6               | Same Sequence Number                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 7               | Logical Node ID                                   | 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual | 8, 9            | Status (MSB, LSB)                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|                 |                                                   | 10, 11          | State (MSB, LSB)<br>If the MSB of the preceding Status field is 0x18 (Invalid Channel B State) or 0x1D (Invalid Channel A State), then this field indicates the component state. If the State value is greater than 0xFF (255), then it is truncated to 0xFF.<br><br>If the MSB of the preceding Status field is neither 0x18 nor 0x1D, then the value in this field is 0x0000.<br><br>If the preceding Status field is a positive acknowledgement, this field does not apply. |
|                 | Number of AEs to follow<br>0x02                   | 12              | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|                 | AEs<br>0x0D Channel A<br>0x0D Channel B           |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

Connect With Pad 0x0003

|   |                                                                                                                                                                                                                                               |
|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | <p>Channel A Pad Value</p> <p>The dB gain/loss adjustment of the signal being transmitted to Channel A.</p> <p>0x00 +3 dB Gain/Loss</p> <p>0x01 0 dB</p> <p>0x02 2 dB</p> <p>0x03 3 dB</p> <p>0x04 4 dB</p> <p>0x05 6 dB</p> <p>0x06 9 dB</p> |
| : | <p>Channel B Pad Value</p> <p>The dB gain/loss adjustment of the signal being transmitted to Channel B.</p> <p>0x00 +3 dB Gain/Loss</p> <p>0x01 0 dB</p> <p>0x02 2 dB</p> <p>0x03 3 dB</p> <p>0x04 4 dB</p> <p>0x05 6 dB</p> <p>0x06 9 dB</p> |
| : | Checksum                                                                                                                                                                                                                                      |

## ConnectionStatusMsg

---

**Type:** EXS SwitchKit API message

**Description** This message is sent by the Low-Level Communicator (LLC) to an application whenever any application connects to or disconnects from the LLC. These notifications go to applications. All applications connect to LLC. By default, LLC will send **ConnectionStatusMsg** messages to all applications. To disable this functionality, set `SK_DISABLE_CSM =1`.

**NOTE:** An application can use this message to determine that a software version mismatch has occurred. In this case, an application receives a **ConnectionStatusMsg** with status 0x06, which indicates there is an invalid connection. The application needs to manually delete the connection, otherwise, a loop will continue. In CSA, you will receive an error message saying the LLC is a different version of software and you should upgrade CSA for it to work.

This feature is also used, for example, to detect whether an application in a redundant pair disconnects. This includes notification of an LLC or SwitchManager switchover. In a redundant LLC set-up, if the primary LLC fails, a **ConnectionStatusMsg** is sent, stating that an application with the LLC name of '0' has disconnected. The '0' corresponds to the primary LLC.

There are two types of applications:

- Special Applications
  - SwitchManger and Redundant LLC (RLLC) are special applications. SwitchManger always connects to the LLC as the application named 1. RLLC always connects to the LLC as the application named 2.
- User Applications
  - User applications, including the Converged Services Administrator, connect to the LLC as applications with names greater than or equal to 10.

In a redundant LLC set-up, if the primary LLC (PLLC) fails, a **ConnectionStatusMsg** is sent, stating that an application with the LLC name of '0' has disconnected. The '0' corresponds to the primary LLC.

By default Switchkit applications, with the wrong version of software, will cause an error when attempting to connect to the LLC. The connection for this faulty application will be made to the LLC, and LLC will then send a **ConnectionStatusMsg** with an error code to all registered applications. All registered Switchkit applications will get a status of `SK_CSM_NoCompat_App_SW (6)` when this occurs.

Once the **ConnectionStatusMsg** had been successfully sent to all registered applications, the LLC will drop the connection, and log the following within the `maintenance_llc.log`:

## ConnectionStatusMsg

```


* Application Connection Failed
*
* Application cannot remain connected to LLC due to incompatible SK API
* software version.
*
* Application software version:08.02.03
* Switchkit software version:08.03.01

```

**Sent by** Application

**C Structure**

```
typedef struct {
 char IPAddress[50];
 UBYTE Status;
 int Name;
 UBYTE reserved72[4];
} SK_ConnectionStatusMsg;
```

**C++ Class**

```
class SKC_ConnectionStatusMsg : public SKC_ToolkitInbound
{
public:
 const char *getIPAddress() const;
 void setIPAddress(const char *x);
 UBYTE getStatus() const;
 void setStatus(UBYTE x);
 int getName() const;
 void setName(int x);
};
```

**Argument Values** The following table shows the possible values for the arguments of this message:

## ConnectionStatusMsg

| <b>Argument</b> | <b>Value</b>                                                                    | <b>Description</b>                                                                                                                                                                                                                                                    |
|-----------------|---------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 | <i>SK_CSM_OtherAppIDDropped</i>                                                 | Indicates the application has disconnected.                                                                                                                                                                                                                           |
|                 | <i>SK_CSM_OtherAppIDConnected</i>                                               | Indicates another application has connected. This value is only received, if the application is registered for the connection status message. Then, the application will get the connection status message for itself and all other applications that connect to LLC. |
| Status          | <i>SK_CSM_AppIDReturnedFromPhantom</i>                                          | This is sent when an application reconnects to an LLC, either because it was disconnected and reconnected within the 15 second time-out period, or there was a switchover from the PLLC to the RLLC.                                                                  |
|                 | <i>SK_CSM_CurrentAppIDConnected</i>                                             | The current application has connected to the LLC/ RLLC.                                                                                                                                                                                                               |
|                 | <i>SK_CSM_AppDropped_NonCompatSoftware</i>                                      | Indicates a SwitchKit user application is trying to connect to an LLC which has a different software version. When a SwitchKit application connects to the LLC, it must have the same software version as the LLC it is connecting to.                                |
| Name            | The LLC name (i.e. integer) associated with the application                     | The LLC name (i.e. integer) associated with the application.                                                                                                                                                                                                          |
| IPAddress       | IP address of the client that the LLC sees the incoming socket connecting from. | IP address of the client that the LLC sees the incoming socket connecting from.                                                                                                                                                                                       |

## CPC Detection 0x0047

---

**SwitchKit Name** CPCDetection

**Type** EXS API and SwitchKit API message

**Description** **CPC Detection 0x0047**

This message enables or disables the detection of CPC (Calling Party Control), which is needed on FXO/LS channels. CPC is the means by which the calling end notifies the called end that the established connection is no longer needed. In this case, dialtone is used for notification.

This feature can only be activated if the CSP contains an MFDSP or DSP-ONE card with at least one DSP configured for Call Progress Analysis. When CPC is detected, it is reported to the host in a *Call Processing Event* message.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
 UBYTE Action;
} XL_CPCDetection;
```

**C++ Class**

```
class XLC_CPCDetection : public XLC_OneChannelOutbound {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 UBYTE getAction() const;
 void setAction(UBYTE x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                                             | RESPONSE (Gray) |                       |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                                                           | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)                                                                                                                                                                                                                                                                                                             | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x0047)                                                                                                                                                                                                                                                                                                       | 3, 4            | Message Type (0x0047) |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                             | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                                                                                                                                                                                             | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                                                                                                                                                                                             | 7               | Logical Node ID       |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs                                                                                                                                                                                                                                                                       | 8, 9            | Status (MSB, LSB)     |
|                 | Number of AEs                                                                                                                                                                                                                                                                                                               | 10              | Checksum              |
|                 | AEs<br>0x0D Channel                                                                                                                                                                                                                                                                                                         |                 |                       |
| :               | Action<br>0x01 Enable CPC Detection<br>0x02 Disable CPC Detection<br>The default time for receiving valid dialtone is 1.5 seconds before the CSP sends the <i>Call Processing Event</i> field value 0x24 (CPC Detected) to the host. To change this value, use the <i>Call Progress Analysis Pattern Configure</i> message. |                 |                       |
| :               | Checksum                                                                                                                                                                                                                                                                                                                    |                 |                       |

# CreateConnection

---

**Type:** EXS SwitchKit API message

**Description** An application that needs to connect to a specific LLC must issue a function call **sk\_createConnection()**. The function then uses the *SK\_CreateConnection* message to establish the connection between the LLC and requesting application.

**Important!** It is not possible to send the CreateConnection message directly to the LLC.

**Sent by** Application

**C Structure**

```
typedef struct {
 UBYTE ConnectionID;
 char PriHost[30];
 char RedHost[30];
 unsigned short PriPort;
 unsigned short RedPort;
 UBYTE Action;
} SK_CreateConnection;
```

**C++ Class**

```
class SKC_CreateConnection : public SKC_ToolkitMessage {
public:
 UBYTE getConnectionID() const;
 void setConnectionID(UBYTE x);
 const char *getPriHost() const;
 void setPriHost(const char *x);
 const char *getRedHost() const;
 void setRedHost(const char *x);
 unsigned short getPriPort() const;
 void setPriPort(unsigned short x);
 unsigned short getRedPort() const;
 void setRedPort(unsigned short x);
 UBYTE getAction() const;
 void setAction(UBYTE x);
};
```

# Cross Connect Channel 0x001A

---

**SwitchKit Name** CrossConnectChannel

**Type** EXS API and SwitchKit API message

**Description** **Cross Connect Channel 0x001A**

This message makes a connection between two channels' data paths. No signaling information can be used to detect either channel releasing. The host must tear the connection down using the *Cross Disconnect Channel* message. If you cross-connect T1 and E1 channels, the result is automatic PCM conversion (A-law/ $\mu$ -law). By default, channels on T1 spans are  $\mu$ -law and channels on E1 spans are A-law. To disable conversion, as would be required if you connect through an ISDN D channel or SS7 link, you must configure both channels with the same format using the *PCM Encoding Format Configure* message.

This message is supported by the VDAC-ONE card but not the IP Network Interface Series 2 card.

**NOTE:** When cross-connecting channels across nodes in an CSP system, a separate message must be sent to each channel in the connection.

For example, you would send one message to Node 1: *Cross Connect Channel A, B*. You would send another message to Node 2: *Cross Connect Channel B, A*.

You can use optional fields for Encoding Format and Pad Value, which are described below the message table. The Encoding Format and Pad Value fields are required for cross-connections involving separate nodes in a multi-node system only.

**Sent by** Host

**SwitchKit Code** **Configuration**

```
CrossConnectChannel (
 Node = integer,
 SpanA = integer,
 ChannelA = integer,
 SpanB = integer,
 ChannelB = integer,
 AEncoding = integer,
 APad = integer,
 BEncoding = integer,
 BPad = integer);
```

**C Structure**

```
typedef struct {
```

---

```

unsigned short SpanA;
UBYTE ChannelA;
unsigned short SpanB;
UBYTE ChannelB;
UBYTE AEncoding;
UBYTE APad;
UBYTE BEncoding;
UBYTE BPad;
} XL_CrossConnectChannel;

```

**C++ Class**

```

class XLC_CrossConnectChannel : public
 XLC_OutboundMessage {
public:
 unsigned short getSpanA() const;
 void setSpanA(unsigned short x);
 UBYTE getChannelA() const;
 void setChannelA(UBYTE x);
 unsigned short getSpanB() const;
 void setSpanB(unsigned short x);
 UBYTE getChannelB() const;
 void setChannelB(UBYTE x);
 UBYTE getAEncoding() const;
 void setAEncoding(UBYTE x);
 UBYTE getAPad() const;
 void setAPad(UBYTE x);
 UBYTE getBEncoding() const;
 void setBEncoding(UBYTE x);
 UBYTE getBPad() const;
 void setBPad(UBYTE x);

```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                       | RESPONSE (Gray) |                       |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                     | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                          | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x0015)                                                                                                                       | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x001A)                                                                                                                 | 3, 4            | Message Type (0x001A) |
| 5               | Reserved (0x00)                                                                                                                       | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                       | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                       | 7               | Logical Node ID       |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs<br><hr/> Number of AEs<br>0x02<br><hr/> AEs<br>0x0D Channel A<br>0x0D Channel B | 8, 9            | Status (MSB, LSB)     |
| :               | Channel A Encoding Format<br>0x01 $\mu$ -law<br>0x02 A-law                                                                            | 10              | Checksum              |

Cross Connect Channel 0x001A

|   |                                                                                                                                                                                                                                                                                                                                                                                                                |
|---|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | <p>Channel A Pad Value</p> <p><b>NOTE:</b> This is a blind connection and should only be done with spans and channels out of service.</p> <p>The dB gain/loss adjustment of the signal being transmitted to the appropriate channel:</p> <p>0x00 +3 dB Gain/Loss<br/>         0x01 0 dB<br/>         0x02 2 dB<br/>         0x03 3 dB<br/>         0x04 4 dB<br/>         0x05 6 dB<br/>         0x06 9 dB</p> |
| : | <p>Channel B Encoding Format</p> <p>0x01 <math>\mu</math>-law<br/>         0x02 A-law</p>                                                                                                                                                                                                                                                                                                                      |
| : | <p>Channel B Pad Value</p> <p><b>NOTE:</b> This is a blind connection and should only be done with spans and channels out of service.</p> <p>The dB gain/loss adjustment of the signal being transmitted to the appropriate channel:</p> <p>0x00 +3 dB Gain/Loss<br/>         0x01 0 dB<br/>         0x02 2 dB<br/>         0x03 3 dB<br/>         0x04 4 dB<br/>         0x05 6 dB<br/>         0x06 9 dB</p> |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                       |

## Cross Connect Span 0x001C

---

**SwitchKit Name** CrossConnectSpan

**Type** EXS API and SwitchKit API message

**Description** **Cross Connect Span 0x001C**

This message connects all data paths on Logical Span ID A to data paths on Logical Span ID B.

When in E1 clear channel mode, this message does not connect time slot 16 (channel 30). You must use the *Cross Connect Channel* message to connect the data paths.

Cross-connection of T1 and E1 spans is not supported. Multiframe alignment does not propagate from one side of the connection to the other.

**NOTE:** When cross-connecting channels across nodes in a multi-node system, a separate message must be sent to each channel in the connection.

For example, you would send one message to Node 1: *Cross Connect Span 0,1*. You would send another message to Node 2: *Cross Connect Span 1,0*. You can use optional fields for Encoding Format and Pad Value, which are described below the message table. The Encoding Format and Pad Value fields are required for cross-connections involving separate nodes in a multi-node CSP system only.

**Sent by** Host

**SwitchKit Code** **Configuration**

```
CrossConnectSpan (
 Node = integer,
 SpanA = integer,
 SpanB = integer,
 ChannelAEncoding = integer,
 ChannelAPad = integer,
 ChannelBEncoding = integer,
 ChannelBPad = integer);
```

**C Structure**

```
typedef struct {
 unsigned short SpanA;
 unsigned short SpanB;
 UBYTE ChannelAEncoding;
 UBYTE ChannelAPad;
 UBYTE ChannelBEncoding;
 UBYTE ChannelBPad;
} XL_CrossConnectSpan;
```

**C++ Class**

```
class XLC_CrossConnectSpan : public XLC_OutboundMessage {
public:
 unsigned short getSpanA() const;
 void setSpanA(unsigned short x);
 unsigned short getSpanB() const;
 void setSpanB(unsigned short x);
 UBYTE getChannelAEncoding() const;
 void setChannelAEncoding(UBYTE x);
 UBYTE getChannelAPad() const;
 void setChannelAPad(UBYTE x);
 UBYTE getChannelBEncoding() const;
 void setChannelBEncoding(UBYTE x);
 UBYTE getChannelBPad() const;
 void setChannelBPad(UBYTE x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                                                     | RESPONSE (Gray) |                            |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|----------------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                                                                   | Byte            | Field Description          |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                        | 0               | Frame (0xFE)               |
| 1, 2            | Length (0x00NN)                                                                                                                                                                                                                                                                                                                     | 1, 2            | Length (0x0007)            |
| 3, 4            | Message Type, MSB (0x001C)                                                                                                                                                                                                                                                                                                          | 3, 4            | Message Type, MSB (0x001C) |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                     | 5               | Reserved (0x00)            |
| 6               | Sequence Number                                                                                                                                                                                                                                                                                                                     | 6               | Same Sequence Number       |
| 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                     | 7               | Logical Node ID            |
| :               | AIB<br>Address Method<br>0x00 - Individual AEs                                                                                                                                                                                                                                                                                      | 8, 9            | Status (MSB, LSB)          |
|                 | Number of AEs to follow                                                                                                                                                                                                                                                                                                             |                 |                            |
|                 | AEs<br>0x0C Logical Span A<br>0x0C Logical Span B                                                                                                                                                                                                                                                                                   |                 |                            |
| :               | Channel A Encoding Format<br>0x01 $\mu$ -law<br>0x02 A-law                                                                                                                                                                                                                                                                          | 10              | Checksum                   |
| :               | Channel A Pad Value<br><b>NOTE:</b> This is a blind connection and should only be done with spans and channels out of service.<br><br>The dB gain/loss adjustment of the signal being transmitted to the appropriate channel:<br>0x00 +3 dB Gain/Loss<br>0x01 0 dB<br>0x02 2 dB<br>0x03 3 dB<br>0x04 4 dB<br>0x05 6 dB<br>0x06 9 dB |                 |                            |

# Cross Connect Span 0x001C

|   |                                                                                                                                                                                                                                                                                                                                     |
|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | Channel B Encoding Format<br>0x01 $\mu$ -law<br>0x02 A-law                                                                                                                                                                                                                                                                          |
| : | Channel B Pad Value<br><b>NOTE:</b> This is a blind connection and should only be done with spans and channels out of service.<br><br>The dB gain/loss adjustment of the signal being transmitted to the appropriate channel:<br>0x00 +3 dB Gain/Loss<br>0x01 0 dB<br>0x02 2 dB<br>0x03 3 dB<br>0x04 4 dB<br>0x05 6 dB<br>0x06 9 dB |
| : | Checksum                                                                                                                                                                                                                                                                                                                            |

# Cross Disconnect Channel 0x001B

---

**SwitchKit Name** CrossDisconnectChannel

**Type** EXS API and SwitchKit API message

**Description** **Cross Disconnect Channel 0x001B**

This message instructs the CSP to disconnect the data paths of the channels specified (for SwitchKit, to disconnect the data paths of the channels that were connected by means of the CrossConnectChannel message).

**NOTE:** When cross-connecting channels across nodes in a CSP system, a separate message must be sent to each channel in the connection.

For example, you would send one message to Node 1:

- *Cross Disconnect Channel* (CrossDisconnectChannel) A, B.

You would send another message to Node 2:

- *Cross Disconnect Channel* (CrossDisconnectChannel) B, A.

**Sent by** Host

**SwitchKit Code** **Configuration**

```
CrossDisconnectChannel (
 Node = integer,
 SpanA = integer,
 ChannelA = integer,
 SpanB = integer,
 ChannelB = integer);
```

**C Structure**

```
typedef struct {
 unsigned short SpanA;
 UBYTE ChannelA;
 unsigned short SpanB;
 UBYTE ChannelB;
} XL_CrossDisconnectChannel;
```

**C++ Class**

```
class XLC_CrossDisconnectChannel : public
 XLC_OutboundMessage {
public:
 unsigned short getSpanA() const;
 void setSpanA(unsigned short x);
 UBYTE getChannelA() const;
 void setChannelA(UBYTE x);
 unsigned short getSpanB() const;
 void setSpanB(unsigned short x);
 UBYTE getChannelB() const;
```

```
void setChannelB(UBYTE x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                | RESPONSE (Gray) |                       |
|-----------------|------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                              | Byte            | Field Description     |
| 0               | Frame (0xFE)                                   | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)                                | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x001B)                          | 3, 4            | Message Type (0x001B) |
| 5               | Reserved (0x00)                                | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                | 7               | Logical Node ID       |
| :               | AIB<br>Address Method<br>0x00 - Individual AEs | 8, 9            | Status (MSB, LSB)     |
|                 | Number of AEs to follow<br>0x02                |                 |                       |
|                 | AEs<br>0x0D Channel A<br>0x0D Channel B        |                 |                       |
| :               | Checksum                                       | 10              | Checksum              |

## Cross Disconnect Span 0x001D

---

**SwitchKit Name** CrossDisconnectSpan

**Type** EXS API and SwitchKit API message

**Description** **Cross Disconnect Span 0x001D**

This message disconnects all data paths of two spans that were previously connected by means of the *Cross Connect Span* message (*XL\_CrossConnectSpan*).

**NOTE:** When cross-connecting channels across nodes in a CSP system, a separate message must be sent to each channel in the connection.

For example, you would send one message to Node 1: *Cross Disconnect Span* (CrossDisconnectSpan) 0,1. You would send another message to Node 2: *Cross Disconnect Span* (CrossDisconnectSpan) 1,0.

**Sent by** Host

**SwitchKit Code** **Configuration**

```
CrossDisconnectSpan (
 Node = integer,
 SpanA = integer,
 SpanB = integer);
```

### C Structure

```
typedef struct {
 unsigned short SpanA;
 unsigned short SpanB;
} XL_CrossDisconnectSpan;
```

### C++ Class

```
class XLC_CrossDisconnectSpan : public
 XLC_OutboundMessage {
public:
 unsigned short getSpanA() const;
 void setSpanA(unsigned short x);
 unsigned short getSpanB() const;
 void setSpanB(unsigned short x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                           | RESPONSE (Gray) |                       |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                         | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                              | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)                                                                                                                                           | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x001D)                                                                                                                                     | 3, 4            | Message Type (0x001D) |
| 5               | Reserved (0x00)                                                                                                                                           | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                           | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                           | 7               | Logical Node ID       |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs<br><hr/> Number of AEs to follow<br>0x02<br><hr/> AEs<br>0x0C Logical Span A<br>0x0C Logical Span B | 8, 9            | Status (MSB, LSB)     |
| :               | Checksum                                                                                                                                                  | 10              | Checksum              |

# CSAPPLConfig

---

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Type</b>           | SwitchKit API message                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Purpose</b>        | This message is used to assign configuration data to the H.323 PPL component, L3P RAS (0x00A0).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Sent by</b>        | Application or SwitchManager                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>SwitchKit Code</b> | <b>Configuration</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|                       | <pre> <b>CSAPPLConfig</b> (     Node = integer,     Range = 0xFFFF:0xFF - 0xFFFF:0xFF,     ComponentID = 0xA0,     Entity = 0x01,     CSASlot = 0xFF,     Protocol = 0x01,     ConfigData = byte array); </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>C Structure</b>    | <pre> typedef struct {     UBYTE AddrInfo[30];     unsigned short ComponentID;     UBYTE Entity;     unsigned short CSASlot;     unsigned short Protocol;     UBYTE ConfigData[216]; } <b><i>XL_CSAPPLConfig</i></b>; </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>C++ Class</b>      | <pre> class <b><i>XLC_CSAPPLConfig</i></b> : XLC_OutboundMessage { public:     const UBYTE *getAddrInfo() const ;     UBYTE *getAddrInfo();     void setAddrInfo(UBYTE *x);     XBYTE getStartSpan() const ;     void setStartSpan(XBYTE x) ;     UBYTE getStartChannel() const;     void setStartChannel(UBYTE x);     XBYTE getEndSpan() const;     void setEndSpan(XBYTE x);     UBYTE getEndChannel() const;     void setEndChannel(UBYTE x);     XBYTE getSpan() const;     void setSpan(XBYTE x);     UBYTE getChannel() const;     void setChannel(UBYTE x);     UBYTE getStackID() const;     void setStackID(UBYTE x);     UBYTE getLinkID() const;     void setLinkID(UBYTE x);     XBYTE getV5ID() const;     void setV5ID(XBYTE x);     XBYTE getRouterHandle() const;     void setRouterHandle(unsigned short x); </pre> |

```
unsigned short GetComponentID() const;
void setComponentID(unsigned short x);
UBYTE getEntity() const;
void setEntity(UBYTE x);
unsigned short getCSASlot() const;
void setCSASlot(unsigned short x);
unsigned short getProtocol() const;
void setProtocol(unsigned short x);
const UBYTE *getConfigData() const;
UBYTE *getConfigData();
void setConfigData(UBYTE *x)
};
```

# CSAPPLTimerConfig

---

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Type</b>           | SwitchKit API message                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Purpose</b>        | This message allows the host to configure PPL timers for the H.323 PPL component, L3P RAS (0x00A0).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Sent by</b>        | Application or SwitchManager                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>SwitchKit Code</b> | <b>Configuration</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|                       | <pre> <b>CSAPPLTimerConfig</b> (     Node = integer,     Range = 0xFFFF:0xFF - 0xFFFF:0xFF,     ComponentID = 0xA0,     Entity = 0x01,     CSASlot = 0xFF,     Protocol = 0x01,     TimerType = 0x01,     TimerID = integer,     TimerValue = integer, ); </pre>                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>C Structure</b>    | <pre> typedef struct {     UBYTE AddrInfo[30];     unsigned short ComponentID;     UBYTE TimerType;     UBYTE TimerID;     unsigned short TimerValue;     unsigned short CSASlot;     unsigned short Protocol;     UBYTE reserved57[213]; } <b><i>XL_CSAPPLTimerConfig</i></b>; </pre>                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>C++ Class</b>      | <pre> class <b><i>XLC_CSAPPLTimerConfig</i></b> : XLC_OutboundMessage { public:     const UBYTE *getAddrInfo() const ;     UBYTE *getAddrInfo();     void setAddrInfo(UBYTE *x);     XBYTE getStartSpan() const ;     void setStartSpan(XBYTE x) ;     UBYTE getStartChannel() const;     void setStartChannel(UBYTE x);     XBYTE getEndSpan() const;     void setEndSpan(XBYTE x);     UBYTE getEndChannel() const;     void setEndChannel(UBYTE x);     XBYTE getSpan() const;     void setSpan(XBYTE x);     UBYTE getChannel() const;     void setChannel(UBYTE x);     UBYTE getStackID() const;     void setStackID(UBYTE x);     UBYTE getLinkID() const; </pre> |

```
void setLinkID(UBYTE x);
XBYTE getV5ID() const;
void setV5ID(XBYTE x);
unsigned short getComponentID() const;
void setComponentID(unsigned short x);
UBYTE getTimerType() const;
void setTimerType(UBYTE x);
UBYTE getTimerID() const;
void setTimerID(UBYTE x);
unsigned short getTimerValue() const;
void setTimerValue(unsigned short x);
unsigned short getCSASlot() const;
void setCSASlot(unsigned short x);
unsigned short getProtocol() const;
void setProtocol(unsigned short x);
};
```

## D Channel Assign 0x00C4

---

|                       |                                                                                                                                                                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | DChannelAssign                                                                                                                                                                                                                      |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                                                                                                                                   |
| <b>Description</b>    | <p><b>D Channel Assign 0x00C4</b></p> <p>This message assigns a channel as an ISDN PRI or an ISDN Series 3 D channel or a subrate channel as an ISDN BRI channel. ISDN channels can be assigned as either primary or secondary.</p> |
| <b>Sent by</b>        | Host                                                                                                                                                                                                                                |

### SwitchKit Code Configuration

In the configuration below you can replace:

```
PriDChannelSpanID = integer,
PriDChannel = integer,
```

with

```
SecondaryDChannelSpanID = integer,
SecondaryDChannel = integer,
```

```
DChannelAssign (
 Node = integer,
 Slot = integer,
 PriDChannelSpanID = integer,
 PriDChannel = integer,
 SecondaryDChannelFacility = integer,
 DChannelType = integer,
```

### C Structure

```
typedef struct {
 UBYTE AddrInfo[30];
 UBYTE SecondaryDChannelFacility;
 UBYTE DChannelType;
 UBYTE Data;
} XL_DChannelAssign;
```

### C++ Class

```
class XLC_DChannelAssign : public XLC_OutboundMessage {
public:
 const UBYTE *getAddrInfo() const;
 UBYTE *getAddrInfo();
 void setAddrInfo(UBYTE *x);
 UBYTE getSecondaryDChannelFacility() const;
 void setSecondaryDChannelFacility(UBYTE x);
 UBYTE getDChannelType() const;
 void setDChannelType(UBYTE x);
 UBYTE getData() const;
 void setData(UBYTE x);
```

};

**EXS API Hex Format**

| MESSAGE (White)                                                                                                                           |                                                                                        | RESPONSE (Gray) |                       |
|-------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte                                                                                                                                      | Field Description                                                                      | Byte            | Field Description     |
| 0                                                                                                                                         | Frame (0xFE)                                                                           | 0               | Frame (0xFE)          |
| 1, 2                                                                                                                                      | Length (0xNNNN)                                                                        | 1, 2            | Length (0x0007)       |
| 3, 4                                                                                                                                      | Message Type (0x00C4)                                                                  | 3, 4            | Message Type (0x00C4) |
| 5                                                                                                                                         | Reserved (0x00)                                                                        | 5               | Reserved (0x00)       |
| 6                                                                                                                                         | Sequence Number                                                                        | 6               | Same Sequence Number  |
| 7                                                                                                                                         | Logical Node ID                                                                        | 7               | Logical Node ID       |
| :                                                                                                                                         | AIB                                                                                    | 8, 9            | Status MSB, ISB       |
|                                                                                                                                           | Address Method<br>0x00 - Individual AEs                                                | 10              | Checksum              |
|                                                                                                                                           | Number of AEs                                                                          |                 |                       |
|                                                                                                                                           | AEs<br>0x15 ISDN Primary D Channel<br>or<br>0x16 ISDN Secondary D Channel              |                 |                       |
| Secondary D Channel Facility (See table below)<br>0x00 Assign primary ISDN PRI D channel<br>0x01-0x09 Assign secondary ISDN PRI D channel |                                                                                        |                 |                       |
| :                                                                                                                                         | D Channel Type<br>0x01 ISDN PRI Primary D channel<br>0x02 ISDN PRI Secondary D channel |                 |                       |
|                                                                                                                                           | Checksum                                                                               |                 |                       |

**Secondary D Channel Facility**

The table below shows timeslots on an E1 span that map to logical channels in the CSP. You must assign D channels to timeslot 16 (Channel 30 (0x1E) as noted with shading). You cannot use Channels 30 and 31 for voice/data when configured for ISDN applications.

D Channel Assign 0x00C4

| <b>Timeslot</b> | <b>Logical Channel</b> |
|-----------------|------------------------|
| 0               | 31                     |
| 1               | 0                      |
| 2               | 1                      |
| 3               | 2                      |
| 4               | 3                      |
| 5               | 4                      |
| 6               | 5                      |
| 7               | 6                      |
| 8               | 7                      |
| 9               | 8                      |
| 10              | 9                      |
| 11              | 10                     |
| 12              | 11                     |
| 13              | 12                     |
| 14              | 13                     |
| 15              | 14                     |
| 16              | 30                     |
| 17              | 15                     |
| 18              | 16                     |
| 19              | 17                     |
| 20              | 18                     |
| 21              | 19                     |
| 22              | 20                     |
| 23              | 21                     |
| 24              | 22                     |
| 25              | 23                     |
| 26              | 24                     |
| 27              | 25                     |
| 28              | 26                     |
| 29              | 27                     |
| 30              | 28                     |
| 31              | 29                     |

## D Channel De-assign 0x00C5

---

**SwitchKit Name** DChannelDeAssign

**Type** EXS API and SwitchKit API message

**Description** **D Channel De-assign 0x00C5**

This message is used by the host to de-assign an ISDN PRI D channel. The D channel may then be re-assigned using the *D Channel Assign* message. D channel De-assign messages must be sent for each of the D channels (DSOs) making up the Super Rate D channel. The base D channel must be de-assigned before any of the others can be de-assigned.

**Sent by** Host

**SwitchKit Code** **Configuration**

```
DChannelDeassign (
 Node = integer,
 Span = integer,
 Channel = integer);
```

**C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
} XL_DChannelDeAssign;
```

**C++ Class**

```
class XLC_DChannelDeAssign : public
 XLC_OneChannelOutbound {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                | RESPONSE (Gray) |                       |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                              | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                   | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)                                                                                                                                | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x00C5)                                                                                                                          | 3, 4            | Message Type (0x00C5) |
| 5               | Reserved (0x00)                                                                                                                                | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                | 7               | Logical Node ID       |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs<br><hr/> Number of AEs to follow<br><hr/> AEs<br>0x0D Channel<br>or<br>0x23 Slot/Subrate | 8, 9            | Status MSB, LSB       |
| :               | Checksum                                                                                                                                       | 10              | Checksum              |

## D Channel Facility List Configure 0x00C6

---

**SwitchKit Name** DChannelFacilityListConfig

**Type** EXS API and SwitchKit API message

**Description** **D Channel Facility List Configure 0x00C6**

This message is used by the host to add or delete spans from the facility list of D channels in NFAS mode. A D channel can control up to 20 spans of NI2 channels and 10 spans for all other channel types. The span where the D channel is assigned is automatically added to the facility list as Facility 0. You do not need to add it to the facility list using this message.

**Sent by** Host

**SwitchKit Code** **Configuration**

```
DChannelFacilityListConfig (
 Node = integer,
 Span = integer,
 Channel = integer,
 FacilitySpan = integer,
 Action = integer,
 FacilityNum = integer);
```

**C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
 unsigned short FacilitySpan;
 UBYTE Action;
 UBYTE FacilityNum;
} XL_DChannelFacilityListConfig;
```

**C++ Class**

```
class XLC_DChannelFacilityListConfig : public
 XLC_OutboundMessage {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 unsigned short getFacilitySpan() const;
 void setFacilitySpan(unsigned short x);
 UBYTE getAction() const;
 void setAction(UBYTE x);
 UBYTE getFacilityNum() const;
 void setFacilityNum(UBYTE x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                             | RESPONSE (Gray) |                       |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                                                                           | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                                                                                | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)                                                                                                                                                                                             | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x00C6)                                                                                                                                                                                       | 3, 4            | Message Type (0x00C6) |
| 5               | Reserved (0x00)                                                                                                                                                                                             | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                                                                             | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                                                                             | 7               | Logical Node ID       |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs<br>Number of AEs to follow<br>AEs<br>0x0D Channel<br>0x0C Logical Span (See below)                                                                    | 8, 9            | Status MSB, LSB       |
| :               | Action<br>0x01 Add facility<br>0x02 Delete facility                                                                                                                                                         | 10              | Checksum              |
| :               | Facility Number<br>0x01–0x09 Index into the Facility List where the span should be deleted or added<br>0x01–0x13 Index into the Facility List where the span should be deleted or added (NI2 channels only) |                 |                       |
| :               | Checksum                                                                                                                                                                                                    |                 |                       |

Use the Channel and Span address type for this message. The table that follows shows you the format for addressing the span for Facility 0. Repeat the same format for facilities 1–9 (0x01–0x09) or 1-19 (0x01–0x13) for NI2 channels only.

| Byte             | AIB Field Description        |
|------------------|------------------------------|
| Channel          |                              |
| 0                | Address Type (0x0D)          |
| 1                | Data Length (0x03)           |
| 2                | Data[0] Logical Span ID, MSB |
| 3                | Data[1] Logical Span ID, LSB |
| 4                | Data[2] Channel              |
| Span, Facility 0 |                              |
| 5                | Address Type (0x0C)          |
| 6                | Data Length (0x02)           |
| 7                | Data[0] Logical Span ID, MSB |
| 8                | Data[1] Logical Span ID, LSB |

## D Channel Facility List Query 0x00CB

---

|                       |                                                                                                                         |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | DChannelFacilityListQuery                                                                                               |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                       |
| <b>Description</b>    | <b>D Channel Facility List Query 0x00CB</b><br>This message is used by the host to request a D channel's Facility List. |
| <b>Sent by</b>        | Host                                                                                                                    |
| <b>SwitchKit Code</b> | <b>C Structure</b>                                                                                                      |

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
} XL_DChannelFacilityListQuery;
```

### C Structure Response

```
typedef struct {
 unsigned short Status;
 unsigned short Span;
 UBYTE Channel;
 UBYTE FacilitySpan0;
 UBYTE FacilitySpan1;
 UBYTE FacilitySpan2;
 UBYTE FacilitySpan3;
 UBYTE FacilitySpan4;
 UBYTE FacilitySpan5;
 UBYTE FacilitySpan6;
 UBYTE FacilitySpan7;
 UBYTE FacilitySpan8;
 UBYTE FacilitySpan9;
} XL_DChannelFacilityListQueryAck;
```

### C++ Class

```
class XLC_DChannelFacilityListQuery : public
 XLC_OneChannelOutbound {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
};
```

### C++ Class Response

```
class XLC_DChannelFacilityListQueryAck : public
 XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 unsigned short getSpan() const;
```

## D Channel Facility List Query 0x00CB

```

void setSpan(unsigned short x);
UBYTE getChannel() const;
void setChannel(UBYTE x);
UBYTE getFacilitySpan0() const;
void setFacilitySpan0(UBYTE x);
UBYTE getFacilitySpan1() const;
void setFacilitySpan1(UBYTE x);
UBYTE getFacilitySpan2() const;
void setFacilitySpan2(UBYTE x);
UBYTE getFacilitySpan3() const;
void setFacilitySpan3(UBYTE x);
UBYTE getFacilitySpan4() const;
void setFacilitySpan4(UBYTE x);
UBYTE getFacilitySpan5() const;
void setFacilitySpan5(UBYTE x);
UBYTE getFacilitySpan6() const;
void setFacilitySpan6(UBYTE x);
UBYTE getFacilitySpan7() const;
void setFacilitySpan7(UBYTE x);
UBYTE getFacilitySpan8() const;
void setFacilitySpan8(UBYTE x);
UBYTE getFacilitySpan9() const;
void setFacilitySpan9(UBYTE x);
};

```

### EXS API Hex Format

| MESSAGE (White) |                                                                                                                                                                                                                                    | RESPONSE (Gray) |                                                       |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-------------------------------------------------------|
| Byte            | Field Description                                                                                                                                                                                                                  | Byte            | Field Description                                     |
| 0               | Frame (0xFE)                                                                                                                                                                                                                       | 0               | Frame (0xFE)                                          |
| 1, 2            | Length (0xNNNN)                                                                                                                                                                                                                    | 1, 2            | Length (0xNNNN)                                       |
| 3, 4            | Message Type (0x00CB)                                                                                                                                                                                                              | 3, 4            | Message Type (0x00CB)                                 |
| 5               | Reserved (0x00)                                                                                                                                                                                                                    | 5               | Reserved (0x00)                                       |
| 6               | Sequence Number                                                                                                                                                                                                                    | 6               | Same Sequence Number                                  |
| 7               | Logical Node ID                                                                                                                                                                                                                    | 7               | Logical Node ID                                       |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs                                                                                                                                                                              | 8, 9            | Status MSB, LSB                                       |
|                 | Number of AEs                                                                                                                                                                                                                      | :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs |
|                 | AE<br>0x0D Channel                                                                                                                                                                                                                 |                 | Number of AEs                                         |
| :               | Checksum                                                                                                                                                                                                                           |                 | AEs<br>0x0D Channel<br>0x0C Logical Span (See below)  |
|                 | Response continued below.                                                                                                                                                                                                          |                 |                                                       |
| :               | Facility 0 Span ID<br>This field represents a span that the specified D channel is controlling. The response contains 10 of these fields, one for each facility (0-9).<br>The value for a facility that is not configured is 0xFF. |                 |                                                       |

D Channel Facility List Query 0x00CB

|   |                    |
|---|--------------------|
| : | :                  |
| : | Facility 9 Span ID |
| : | Checksum           |

The Channel and Span address type is used for this message. The table that follows shows you the format for addressing the span for Facility 0. Repeat the same format for facilities 1–8.

| Byte             | AIB Field Description         |
|------------------|-------------------------------|
| 0                | Address Method (0x00)         |
| 1                | No of Address Elements (0x0A) |
| Channel AE:      |                               |
| 2                | Address Type (0x0D)           |
| 3                | Data Length (0x03)            |
| 4                | Data[0] Logical Span ID, MSB  |
| 5                | Data[1] Logical Span ID, LSB  |
| 6                | Data[2] Channel               |
| Span, Facility 0 |                               |
| 7                | Address Type (0x0C)           |
| 8                | Data Length (0x02)            |
| 9                | Data[0] Logical Span ID, MSB  |
| 10               | Data[1] Logical Span ID, LSB  |

## DeviceServerEx

---

|                    |                                                                                                                                                                                                                                    |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Type</b>        | SwitchKit API message                                                                                                                                                                                                              |
| <b>Purpose</b>     | Use the <i>SK_DeviceServerEx</i> message to configure the H.323 IP Signaling Series 3 card.                                                                                                                                        |
| <b>Description</b> | This message is used by the CSA to configure the H.323 IP Signaling Series 3 card. Its configuration syntax corresponds to the message fields of the <i>XL_DeviceServerConfigure</i> and <i>XL_IPCallServerConfigure</i> messages. |
| <b>Sent by</b>     | Host, via CSA configuration or SwitchManager configuration file.                                                                                                                                                                   |

**Arguments** The following table shows the arguments you can change:

| Arguments            | Description                                                                          |
|----------------------|--------------------------------------------------------------------------------------|
| ObjectType           | 0x0020                                                                               |
| ID                   | Device Server ID                                                                     |
| CallServerID         | The LNI of the CSP Matrix Series 3 Card                                              |
| Slot                 | Slot for the Device Server                                                           |
| ConfigTag            | ignored by SMgr                                                                      |
| CompatFlat           | any number                                                                           |
| PollInterval         | see the ServerConfig message                                                         |
| AllowedMissedPolls   | see the ServerConfig message                                                         |
| Port                 | Device Server IP Port                                                                |
| PrimaryCSPort        | Port for CSP Matrix Series 3 Card to listen on                                       |
| SecondaryCSPort      | Port for CSP Matrix Series 3 Card to listen on                                       |
| IpAddress            | IP at Device Server                                                                  |
| PrimaryCSIpAddress   | IP at CSP Matrix Series 3 Card                                                       |
| SecondaryCSIpAddress | IP at CSP Matrix Series 3 Card                                                       |
| DeviceType           | 0x01                                                                                 |
| Platform             | See the DeviceServerConfig message by clicking below:<br><br>Matrix Configure 0x007D |
| IPDCUsage            | See the DeviceServerConfig message by clicking below:<br><br>Matrix Configure 0x007D |
| DSInterfaceSupport   | See the DeviceServerConfig message by clicking below:<br><br>Matrix Configure 0x007D |

**Configuration** *DeviceServerEx* (  
 ObjectType = integer,  
 Id = integer,

## DeviceServerEx

```
CallServerId = integer,
Slot = integer,
ConfigTag = integer,
CompatFlag = integer,
PollInterval = integer,
AllowedMissedPolls = integer,
Port = integer,
PrimaryCSPort = integer,
SecondaryCSPort = integer,
IpAddress = string,
PrimaryCSIpAddress = string,
SecondaryCSIpAddress = string,
DeviceType = integer,
Platform = integer,
IDDCUsage = integer,
DSInterfaceSupport = BYTE ARRAY);
```

## Diagnostics Indication 0x0045

---

**SwitchKit Name**    DiagnosticsIndication

**Type**            EXS API and SwitchKit API message

**Description**    **Diagnostics Indication 0x0045**

This message sends information to the host about a specific module to help diagnose protocol/system problems. Use the *ISDN Interface Configure* message options bitmask entity for enabling/disabling this message.



### **WARNING**

*This message is for debugging purposes only. It should not be sent during live call processing. Use the ISDN Interface Configure message to enable/disable diagnostics.*

**Sent by**        CSP

**SwitchKit Code**    **C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
 UBYTE DiagnosticsType;
 UBYTE Data[222];
} XL_DiagnosticsIndication;
```

### **C++ Class**

```
class XLC_DiagnosticsIndication : public
 XLC_OneChannelMessage {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 UBYTE getDiagnosticsType() const;
 void setDiagnosticsType(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                          | RESPONSE (Gray) |                       |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                                                                                        | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                                                                                             | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)                                                                                                                                                                                                          | 1, 2            | Length (0x0008)       |
| 3, 4            | Message Type (0x0045)                                                                                                                                                                                                    | 3, 4            | Message Type (0x0045) |
| 5               | Reserved (0x00)                                                                                                                                                                                                          | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                                                                                          | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                                                                                          | 7               | Logical Node ID       |
| :               | AIB                                                                                                                                                                                                                      | 8, 9            | Status (MSB, LSB)     |
|                 | Address Method<br>0x00 - Individual AEs                                                                                                                                                                                  |                 |                       |
|                 | Number of AEs                                                                                                                                                                                                            | 11              | Checksum              |
|                 | AEs<br>0x0D Channel                                                                                                                                                                                                      |                 |                       |
| :               | Diagnostics Type<br>0x01 ISDN PRI Cards Raw Layer 3 HDLC frames<br>Data[0]Direction (0= Layer 2 to Layer 3, 1= Layer 3 to Layer 2)<br>Data[1]Length<br>Data[2]Start of the HDLC data<br>:<br>Data[n]End of the HDLC data |                 |                       |
| :               | Data                                                                                                                                                                                                                     |                 |                       |
| :               | Checksum                                                                                                                                                                                                                 |                 |                       |

# Disconnect Tone Pattern 0x001E

---

**SwitchKit Name** DisconnectTonePattern

**Type** EXS API and SwitchKit API message

**Description** **Disconnect Tone Pattern 0x001E**

This message instructs the CSP to terminate call progress tone transmission on the specified channel.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
} XL_DisconnectTonePattern;
```

**C++ Class**

```
class XLC_DisconnectTonePattern : public
 XLC_OneChannelOutbound {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
};
```

## EXS API Hex Format

| MESSAGE (White) |                                                                                                 | RESPONSE (Gray) |                       |
|-----------------|-------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                               | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                    | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)                                                                                 | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x001E)                                                                           | 3, 4            | Message Type (0x001E) |
| 5               | Reserved (0x00)                                                                                 | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                 | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                 | 7               | Logical Node ID       |
| :               | AIB<br>Address Method<br>0x00 - Individual AEs<br>Number of AEs to follow<br>AE<br>0x0D Channel | 8, 9            | Status (MSB, LSB)     |
| :               | Checksum                                                                                        | 10              | Checksum              |

## Distant End Release Mode 0x00B8

---

|                       |                                                                                                                                                                                                                |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | DistantReleaseConfig                                                                                                                                                                                           |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                                                                                                              |
| <b>Description</b>    | <p><b>Distant End Release Mode 0x00B8</b></p> <p>This message sets a channel's distant end release mode, which determines whether the distant end is released or parked when the connection is terminated.</p> |
| <b>Sent by</b>        | Host                                                                                                                                                                                                           |

### SwitchKit Code Configuration

```
DistantReleaseConfig (
 Node = integer,
 Range = StartSpan:StartChan - EndSpan:EndChan,
 ReleaseMode = integer);
```

### C Structure

```
typedef struct {
 unsigned short StartSpan;
 UBYTE StartChannel;
 unsigned short EndSpan;
 UBYTE EndChannel;
 UBYTE ReleaseMode;
} XL_DistantReleaseConfig;
```

### C++ Class

```
class XLC_DistantReleaseConfig : public
 XLC_ChanRangeMessage {
public:
 unsigned short getStartSpan() const;
 void setStartSpan(unsigned short x);
 UBYTE getStartChannel() const;
 void setStartChannel(UBYTE x);
 unsigned short getEndSpan() const;
 void setEndSpan(unsigned short x);
 UBYTE getEndChannel() const;
 void setEndChannel(UBYTE x);
 UBYTE getReleaseMode() const;
 void setReleaseMode(UBYTE x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                        | RESPONSE (Gray) |                       |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                      | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                           | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)                                                                                                                                        | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x00B8)                                                                                                                                  | 3, 4            | Message Type (0x00B8) |
| 5               | Reserved (0x00)                                                                                                                                        | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                        | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                        | 7               | Logical Node ID       |
| :               | <u>AIB</u><br>Address Method<br>0x01 - Range of AEs<br><hr/> Number of AEs to follow<br>0x02<br><hr/> 0x0D Channel (Starting)<br>0x0D Channel (Ending) | 8, 9            | Status MSB, LSB       |
| :               | Release Mode<br>0x01 Park<br>0x02 Release (Default)                                                                                                    | 10              | Checksum              |
| :               | Checksum                                                                                                                                               |                 |                       |

## Download Begin BRecord 0x009B

---

**SwitchKit Name** DownloadBeginBRecord

**Type** EXS API and SwitchKit API message

**Description** **Download Begin BRecord 0x009B**

This message should not be sent by the host to standby CSP 2000 Matrix Cards, since the CSP Matrix Series 3 CardCSP Matrix Series 3 Cards receive their loads from the TFTP download.

However, you can use this message to configure active CSP 2000 Matrix Cards.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 } XLC_DownloadBeginBRecord;
```

**C++ Class**

```
class XLC_DownloadBeginBRecord : public
 XLC_OutboundMessage {
public:
 };
```

**EXS API Hex Format**

| MESSAGE (White) |                       | RESPONSE (Gray) |                       |
|-----------------|-----------------------|-----------------|-----------------------|
| Byte            | Field Description     | Byte            | Field Description     |
| 0               | Frame (0xFE)          | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)       | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x009B) | 3, 4            | Message Type (0x009B) |
| 5               | Reserved (0x00)       | 5               | Reserved (0x00)       |
| 6               | Sequence Number       | 6               | Same Sequence Number  |
| 7               | Logical Node ID       | 7               | Logical Node ID       |
| 8               | Checksum              | 8, 9            | Status MSB, LSB       |
|                 |                       | 10              | Checksum              |

## Download Begin SRecord 0x00A2

---

**SwitchKit Name** DownloadBeginSRecord

**Type** EXS API and SwitchKit API message

**Description** **Download Begin SRecord 0x00A2**

This message should not be sent by the host to standby CSP 2000 Matrix Cards, since the CSP 2000 Matrix Cards receive their loads from the TFTP download.

However, you can use this message to configure active CSP 2000 Matrix Cards.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 } XL_DownloadBeginSRecord;
```

**C++ Class**

```
class XLC_DownloadBeginSRecord : public
 XLC_OutboundMessage {
public:
 };
```

**EXS API Hex Format**

| MESSAGE (White) |                       | RESPONSE (Gray) |                       |
|-----------------|-----------------------|-----------------|-----------------------|
| Byte            | Field Description     | Byte            | Field Description     |
| 0               | Frame (0xFE)          | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)       | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x00A2) | 3, 4            | Message Type (0x00A2) |
| 5               | Reserved (0x00)       | 5               | Reserved (0x00)       |
| 6               | Sequence Number       | 6               | Same Sequence Number  |
| 7               | Logical Node ID       | 7               | Logical Node ID       |
| 8               | Checksum              | 8, 9            | Status MSB, ISB       |
|                 |                       | 10              | Checksum              |

# Download BRecord 0x009C

---

**SwitchKit Name** DownloadBRecord

**Type** EXS API and SwitchKit API message

**Description** **Download BRecord 0x009C**

This message should not be sent by the host to standby CSP 2000 Matrix Cards, since the CSP Matrix Series 3 Cards receive their loads from the TFTP download.

However, you can use this message to configure active CSP 2000 Matrix Cards.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 UBYTE Data[253];
} XL_DownloadBRecord;
```

**C++ Class**

```
class XLC_DownloadBRecord : public XLC_OutboundMessage {
public:
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                             | RESPONSE (Gray) |                       |
|-----------------|-----------------------------|-----------------|-----------------------|
| Byte            | Field Description           | Byte            | Field Description     |
| 0               | Frame (0xFE)                | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)             | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x009C)       | 3, 4            | Message Type (0x009C) |
| 5               | Reserved (0x00)             | 5               | Reserved (0x00)       |
| 6               | Sequence Number             | 6               | Same Sequence Number  |
| 7               | Logical Node ID             | 7               | Logical Node ID       |
| 8               | Number of BRecord Bytes (n) | 8, 9            | Status MSB, LSB       |
| 9               | BRecord Byte[0]             | 10              | Checksum              |
| :               | BRecord Byte[n]             |                 |                       |
| :               | Checksum                    |                 |                       |

# Download Complete 0x00A4

---

**SwitchKit Name** DownloadComplete

**Type** EXS API and SwitchKit API message

**Description** **Download Complete 0x00A4**

This message should not be sent by the host to standby CSP 2000 Matrix Cards, since the CSP Matrix Series 3 Cards receive their loads from the TFTP download.

However, you can use this message to configure active CSP 2000 Matrix Cards.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 int Timestamp;
} XL_DownloadComplete;
```

**C++ Class**

```
class XLC_DownloadComplete : public XLC_OutboundMessage {
public:
 int getTimestamp() const;
 void setTimestamp(int x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                       | RESPONSE (Gray) |                       |
|-----------------|-----------------------|-----------------|-----------------------|
| Byte            | Field Description     | Byte            | Field Description     |
| 0               | Frame (0xFE)          | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)       | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x00A4) | 3, 4            | Message Type (0x00A4) |
| 5               | Reserved (0x00)       | 5               | Reserved (0x00)       |
| 6               | Sequence Number       | 6               | Same Sequence Number  |
| 7               | Logical Node ID       | 7               | Logical Node ID       |
| 8-11            | Timestamp (See below) | 8, 9            | Status MSB, LSB       |
| 12              | Checksum              | 10              | Checksum              |

**Timestamp** This field is a long word received Most Significant Byte first, and Least Significant Byte last. It is associated with the newly downloaded system software once it has been validated.

This field is used to determine which matrix has the most recent system software in a redundant system when two different system software versions

are present. In this situation the system software with the highest timestamp will be transferred to the other matrix.

As far as the matrices are concerned, the timestamp has nothing to do with time, but it is suggested that the host use a time-related number in this field. The host must guarantee that each new load has a higher timestamp than the previous one or the new load will be overwritten with the system software from the other matrix. The format of the timestamp is number of seconds since January 1, 1970.

# Download SRecord 0x00A3

---

**SwitchKit Name** DownloadSRecord

**Type** EXS API and SwitchKit API message

**Description** **Download SRecord 0x00A3**

This message should not be sent by the host to standby CSP 2000 Matrix Cards, since the CSP Matrix Series 3 Cards receive their loads from the TFTP download.

However, you can use this message to configure active CSP 2000 Matrix Cards.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 UBYTE Data[253];
} XL_DownloadSRecord;
```

**C++ Class**

```
class XLC_DownloadSRecord : public XLC_OutboundMessage {
public:
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                             | RESPONSE (Gray) |                       |
|-----------------|-----------------------------|-----------------|-----------------------|
| Byte            | Field Description           | Byte            | Field Description     |
| 0               | Frame (0xFE)                | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)             | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x00A3)       | 3, 4            | Message Type (0x00A3) |
| 5               | Reserved (0x00)             | 5               | Reserved (0x00)       |
| 6               | Sequence Number             | 6               | Same Sequence Number  |
| 7               | Logical Node ID             | 7               | Logical Node ID       |
| 8               | Number of SRecord Bytes (n) | 8, 9            | Status MSB, LSB       |
| 9               | SRecord Byte[0]             |                 |                       |
| :               | :                           | 10              | Checksum              |
| :               | SRecord Byte[n]             |                 |                       |
| :               | Checksum                    |                 |                       |

## DS0 Status Change 0x0042

---

|                       |                                                                                                                                                                                                                                                                                              |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | DS0StatusChange                                                                                                                                                                                                                                                                              |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                                                                                                                                                                                            |
| <b>Description</b>    | <p><b>DS0 Status Change 0x0042</b></p> <p>This message informs the host of the current status of the channel specified. It is sent when the status of a channel has changed. All host applications must be designed to handle this message for out-of-service and in-service conditions.</p> |
| <b>Sent by</b>        | CSP                                                                                                                                                                                                                                                                                          |
| <b>Resent</b>         | This message is resent once after five seconds.                                                                                                                                                                                                                                              |

### SwitchKit Code C Structure

```
typedef struct {
 BaseFields Base;
 UBYTE AddrInfo[30];
 UBYTE ChannelStatus;
 UBYTE PurgeStatus;
} XL_DS0StatusChange;
```

### C++ Class

```
class XLC_DS0StatusChange : public XLC_InboundMessage {
public:
 const UBYTE *getAddrInfo() const;
 UBYTE *getAddrInfo();
 void setAddrInfo(UBYTE *x);
 XBYTE getStartSpan() const;
 void setStartSpan(XBYTE x);
 UBYTE getStartChannel() const;
 void setStartChannel(UBYTE x);
 XBYTE getEndSpan() const;
 void setEndSpan(XBYTE x);
 UBYTE getEndChannel() const;
 void setEndChannel(UBYTE x);
 XBYTE getSpan() const;
 void setSpan(XBYTE x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 UBYTE getChannelStatus() const;
 void setChannelStatus(UBYTE x);
 UBYTE getPurgeStatus() const;
 void setPurgeStatus(UBYTE x);
};
```

### Tip for C Programmers

To extract the span and channel in the C version of the API, you need to use the following code:

```

CASE_DS0StatusChange(ds0) {
 printf("DS0 Stat Change, Span:%d, Chan:%d, Purge:%X\n",
 sk_getAIBSpan(ds0->AddrInfo),
 sk_getAIBChannel(ds0->AddrInfo),
 ds0->ChannelStatus,
 ds0->PurgeStatus);
 return OK;
}

```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                             | RESPONSE (Gray) |                       |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                                                                                           | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                                                                                                | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)                                                                                                                                                                                                             | 1, 2            | Length (0x0005)       |
| 3, 4            | Message Type (0x0042)                                                                                                                                                                                                       | 3, 4            | Message Type (0x0042) |
| 5               | Reserved (0x00)                                                                                                                                                                                                             | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                                                                                             | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                                                                                             | 7               | Logical Node ID       |
| :               | AIB<br>Address Method<br>0x00 - Individual AEs<br><br>or for Line Card Busy:<br><br>0x01 - Range of AEs<br>Number of AEs to follow<br>AEs<br>0x0D Channel<br><br>or<br><br>0x0D Channel (Starting)<br>0x0D Channel (Ending) | 8               | Checksum              |
| :               | Channel Status<br>0x00 Purge<br>0x01 Out of Service<br>0x02 In Service<br>0x03 Reserved<br>0x04 Parked<br>0x05 Blocked<br>0x06 Unblocked<br>0x07 Out-of-Service Maintenance Loopback                                        |                 |                       |
| :               | Purge Reason (See table below)                                                                                                                                                                                              |                 |                       |
| :               | Checksum                                                                                                                                                                                                                    |                 |                       |

**Purge Reason** If the *Channel Status* is 0x00 (Purge), consult the table below for the Purge Reason and corrective action. Otherwise, this field is 0x00.

| Value | Description                     | Comments                                                                                                                                                                                                               | Corrective User Action                                                                                                                         |
|-------|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x01  | Reserved                        |                                                                                                                                                                                                                        |                                                                                                                                                |
| 0x02  | Configuration Non-dormant       | Reserved                                                                                                                                                                                                               | Contact Dialogic Tech Support                                                                                                                  |
| 0x03  | Reserved                        | Reserved                                                                                                                                                                                                               | Contact Dialogic Tech Support                                                                                                                  |
| 0x04  | Disabled While Non-dormant      | Reserved                                                                                                                                                                                                               | Contact Dialogic Tech Support                                                                                                                  |
| 0x05  | Internal Outseize ACK Timeout   | Channel B did not respond to connect request from Channel A within 30 seconds.                                                                                                                                         | None—the CSP has no control over the remote channel.                                                                                           |
| 0x06  | Internal Disconnect ACK Timeout | Channel B did not respond to a disconnect request from Channel A within 6 seconds.                                                                                                                                     | None—the CSP has no control over the remote channel.                                                                                           |
| 0x07  | Internal Connect ACK Timeout    | Reserved                                                                                                                                                                                                               | Contact Dialogic Tech Support                                                                                                                  |
| 0x0A  | Invalid RCM Stack               | The RCM stack is either full or in an incorrect state.                                                                                                                                                                 | Contact Dialogic Tech Support                                                                                                                  |
| 0x0B  | FEM Outseize ACK Timeout        | Layer 4 did not receive an <i>Outseize Control</i> ACK from Layer 3 within 29 seconds.                                                                                                                                 | If using PPL for L3, make sure the <i>Outseize Control</i> ACK is sent from Layer 3 to Layer 4; otherwise, contact Dialogic Technical Support. |
| 0x0C  | Internal Inconsistency          | The RCM stack is in a state inappropriate for the action being processed.                                                                                                                                              | Contact Dialogic Tech Support                                                                                                                  |
| 0x0D  | L3 Clear ACK Timeout            | Layer 3 did not respond to a clear request within 150 seconds.                                                                                                                                                         | If using PPL for L3, make sure that a Clear ACK is sent from L3 to L4; otherwise, contact Dialogic Technical Support.                          |
| 0x0E  | L4 Hold ACK Wait Timeout        | Reserved                                                                                                                                                                                                               | Contact Dialogic Technical Support.                                                                                                            |
| 0x0F  | L3 Answer Wait Timeout          | Layer 4 timed-out waiting for a L3 answer event.<br><br>Default timer: 10 minutes                                                                                                                                      | If using PPL for L3, make sure a CONNECT is sent from L3 to L4. Otherwise, contact Dialogic Technical Support.                                 |
| 0x10  | L4 Answer Wait Timeout          | Layer 4 timed-out while waiting for the other party involved in the connection to propagate an answer.<br><br>Default timer: 10 minutes                                                                                | Verify answer supervision mode.                                                                                                                |
| 0x11  | L4 Recall Wait Timeout          | Timed-out while trying to recall the remote RCM.                                                                                                                                                                       | Contact Dialogic Tech Support                                                                                                                  |
| 0x12  | L3 Channel Out-of-Service       | Channel A did not respond to a Channel Release Request from Channel B. L4 purges in order to bring the purging timeslot into a known state.                                                                            | None, the CSP has no control over the remote channel.                                                                                          |
| 0x13  | Implicated by L3                | Reserved.                                                                                                                                                                                                              | Contact Dialogic Tech Support                                                                                                                  |
| 0x14  | Implicated by L4                | One of the terminating channels has detected an internal error and has been disabled. It also had its status changed to dead and has been reconfigured to some non-idle state or has been implicated in another purge. | Contact Dialogic Tech Support.                                                                                                                 |

| Value | Description                                                                         | Comments                                                                                                                                                                                                                | Corrective User Action                                                                                                                                                         |
|-------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x15  | Implicated by LCM                                                                   | One of the terminating channels has detected an internal error and has been disabled. It also had its status changed to dead and has been reconfigured to some non-idle state, or has been implicated in another purge. | Contact Dialogic Tech Support                                                                                                                                                  |
| 0x16  | Unknown LSM State                                                                   | Reserved.                                                                                                                                                                                                               | Contact Dialogic Tech Support                                                                                                                                                  |
| 0x17  | L3 Pre-Answer Disconnect Request                                                    | Reserved.                                                                                                                                                                                                               | Contact Dialogic Tech Support                                                                                                                                                  |
| 0x18  | L3 Not Idle at Switchover                                                           | Layer 4 is idle and Layer 3 is not idle at switchover. This brings the channel into a known state.                                                                                                                      | Make sure Layer 3 is idle before issuing a switchover.                                                                                                                         |
| 0x19  | Outpulsed Digits Timeout                                                            | L4 went out of service.                                                                                                                                                                                                 | Contact Dialogic Tech Support                                                                                                                                                  |
| 0x1A  | Channel bearer service modification request failed.                                 | Two possible reasons:<br><br>The VDAC-ONE card/IP Network Interface Card is not responding to the modification request.<br><br>The network is not responding to the modification request.                               | Check if the end-point signaling with the CSP properly acknowledged the modification request. (For example, a RE-INVITE in SIP networks.)<br><br>Contact Dialogic Tech Support |
| 0x1B  | Channel bearer service modification request timed out.                              | Layer 4 did not receive any response to modify channel bearer service request from Layer 3 within the allowed time.                                                                                                     | Check if the end-point signaling with the CSP properly acknowledged the modification request ((For example, a RE-INVITE in SIP networks.)<br><br>Contact Dialogic Tech Support |
| 0x1C  | Request to retrieve a physical span/channel resource denied by the internal router. | The virtual span/channel's request to retrieve a physical span/channel denied by the internal router.                                                                                                                   | Check if the resource table configuration with internal router is proper.<br><br>Contact Dialogic Tech Support                                                                 |
| 01D   | Request to retrieve a physical span/channel resource timed out.                     | Virtual span/channel's request to retrieve a physical span/channel timed out. No response from the internal router within the allowed time.                                                                             | Contact Dialogic Tech Support                                                                                                                                                  |
| 0x1E  | Idle Purge                                                                          | L4 went out of service.                                                                                                                                                                                                 | Contact Dialogic Tech Support                                                                                                                                                  |
| 0x1F  | BCC Wait Timeout                                                                    | A create broadcast message was sent to from L4 to BCC trying to create a conference and a timeout occurred waiting for an acknowledgment.<br><br>Default timer: 6 seconds                                               | Send a <i>Release With Data</i> message from the host; otherwise, contact Dialogic Technical Support.                                                                          |

| Value | Description                       | Comments                                                                                                                                                                                        | Corrective User Action                                                                                                                                                  |
|-------|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x20  | L5 Release Data Wait Timeout      | An ISDN <i>Channel Release Request</i> message was sent to the host and a timeout occurred while waiting for a <i>Release With Data</i> message.<br><br>Default timer: 20 seconds               | Send a <i>Release With Data</i> message from the host within 20 seconds.                                                                                                |
| 0x21  | L3 Remotely Blocked               | Layer 4 (L4) purges because it is in a state not ready to receive an Out-of-Service blocked event from the associated Layer 3 channel. This results in bringing the channel into a known state. | None—the CSP has no control over the remote channel.                                                                                                                    |
| 0x22  | DSP Resource Wait Timeout         | Layer 4 purges because a timeout occurred while waiting for DSP resources.                                                                                                                      | Make sure DSP resources are available.                                                                                                                                  |
| 0x23  | Outpulsing Digits Timeout         | Layer 4 purges because a timeout occurred while waiting for the completion of outpulsing digits.                                                                                                | Make sure outpulsing digits are available.                                                                                                                              |
| 0x24  | Ring Inaccessible                 | The node cannot access the EXNET® ring due to a local EXNET® hardware problem.                                                                                                                  | Rectify the hardware problem.                                                                                                                                           |
| 0x25  | Node Inaccessible                 | The remote node cannot be accessed on the EXNET® ring due to an EXNET® hardware problem on the remote node.                                                                                     | Rectify the hardware problem.                                                                                                                                           |
| 0x26  | MCC Response Timeout              | Channel purges because timer expires before MCC sends Layer 4 a response to the <i>Connect to Conference</i> or <i>Connect One-Way to Conference</i> message.<br><br>Default timer: 12 seconds  | Contact Dialogic Tech Support                                                                                                                                           |
| 0x27  | Internal router look up failure   | Internal routing look up for a physical resource negatively acknowledged. Possibly due to lack of free physical resources. That is, demand for physical resources has exceeded the supply.      | Check for consistency between actual physical resources and the internal router database configuration. If no such inconsistency is found, add more physical resources. |
| 0x28  | Internal router look up timed out | Internal routing look up for a physical resource timed out.                                                                                                                                     | Contact Dialogic Tech Support                                                                                                                                           |
| 0x38  | Remote Conferencing Timeout       |                                                                                                                                                                                                 |                                                                                                                                                                         |
| 0x71  | Implicated in Error Detection     | Reserved                                                                                                                                                                                        | Contact Dialogic Tech Support                                                                                                                                           |
| 0x72  | Configured, Non-Dormant           | Reserved                                                                                                                                                                                        | Contact Dialogic Tech Support                                                                                                                                           |
| 0x73  | Span Receiving CGA Alarm          | Channel purged because span received Carrier Group Alarm (CGA).                                                                                                                                 | None—the CSP has no control over the remote channel.                                                                                                                    |
| 0x74  | Disabled While Non-dormant        | Layer 3 purges because it is in a state not ready to bring a channel out-of-service.                                                                                                            | Do not take a channel out-of-service when it is involved in a connection.                                                                                               |
|       |                                   | Sent to host whenever an E1 PPL channel is taken out-of-service with the <i>Service State Configure</i> message.                                                                                | No corrective action is necessary.                                                                                                                                      |

| Value | Description                                            | Comments                                                                                                                                                                                       | Corrective User Action                                                                                                                                                                |
|-------|--------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x75  | Internal Outseize ACK Timeout                          | Layer 3 purges because it did not receive a wink when attempting to busy out a channel.                                                                                                        | Verify outgoing start dial type.                                                                                                                                                      |
| 0x76  | Internal Disconnect Clear Wait Timeout                 | Layer 3 purges because a timeout occurred while waiting for Layer 4 to be cleared of a call.                                                                                                   | Contact Dialogic Tech Support                                                                                                                                                         |
| 0x77  | Internal Connect Wait Timeout                          | Layer 3 purges because a timeout occurred while waiting for the channel to be connected.                                                                                                       | None—the CSP has no control over the remote channel.                                                                                                                                  |
| 0x78  | Busy Out ACK Timeout                                   | Layer 3 purges because a time-out occurred while waiting for a busy out acknowledgment from Layer 4.                                                                                           | Contact Dialogic Tech Support                                                                                                                                                         |
| 0x79  | Internal Inconsistency                                 | Layer 3 purges with an unexpected internal error.                                                                                                                                              | Contact Dialogic Tech Support                                                                                                                                                         |
| 0x7A  | Layer 4-implicated Layer 3 Purge                       | Layer 4 purge implication message based upon the current condition of the call stack as well as the associated Layer 3 timeslot.                                                               | Contact Dialogic Tech Support                                                                                                                                                         |
| 0x7B  | CFG-implicated Layer 3 Purge                           | A channel configuration has purged to allow a new configuration to be assigned.                                                                                                                | None—this is not an error.                                                                                                                                                            |
| 0x7D  | Null <i>Outseize Control</i> Instruction Encountered   | A null outseize instruction was encountered.                                                                                                                                                   | Modify <i>Outseize Control</i> message.                                                                                                                                               |
| 0x7E  | Null Outseize Instruction List Instruction Encountered | A null outseize instruction list instruction was encountered.                                                                                                                                  | Modify outseize instruction list.                                                                                                                                                     |
| 0x7F  | Invalid Outseize Instruction Encountered               | An invalid outseize instruction was encountered.                                                                                                                                               | Modify <i>Outseize Control</i> message or programmed outseize instruction list.                                                                                                       |
| 0x80  | Invalid Outseize Instruction Data                      | An invalid wink number has been specified in either:<br><br>– A Scan for Wink N ICB in an <i>Outseize Control</i> message.<br><br>– A preprogrammed Scan For Wink N outseize instruction.      | Modify ICB in <i>Outseize Control</i> message or instruction data byte 1 in the preprogrammed outseize instruction list. The wink number must be from 1 to 8.                         |
| 0x81  | No Outseize Data Available                             | An Outpulse Stage N Address Data instruction has been included in either an <i>Outseize Control</i> message or preprogrammed outseize instructions without corresponding Stage N Address Data. | Include a Stage N Address Data ICB in the <i>Outseize Control</i> message or instruction data in the programmed instructions.                                                         |
| 0x82  | No Outseize ACK Outstanding                            | A Send Host ACK instruction was included in either an <i>Outseize Control</i> message or a programmed outseize instruction list and there are no outstanding outseize instructions.            | Make sure there is only one Send Host ACK instruction per <i>Outseize Control</i> or outseize instruction list.                                                                       |
| 0x83  | Unexpected Off-hook Detected                           | Layer 3 is in a “Wait For Host State” where answer is not allowed.                                                                                                                             | Modify the <i>Outseize Control</i> message or programmed outseize instruction list to use a Wait for Host Control with Answer Supervision ICB instead of a Wait For Host Control ICB. |

| Value | Description                                      | Comments                                                                                                                                                                                                                                                 | Corrective User Action                                                                                                                                                          |
|-------|--------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x84  | On-hook Host Control Wait Timeout                | Layer 3 purged waiting for host control within 120 seconds.                                                                                                                                                                                              | Provide host control within 120 seconds.                                                                                                                                        |
| 0x85  | On-hook Host Data Wait Timeout                   | An Outpulse Stage N Address Data instruction has been included in either an <i>Outseize Control</i> message or preprogrammed outseize instructions without corresponding Stage N Address Data within the timer period.                                   | Include a Stage N Address Data ICB in the <i>Outseize Control</i> message or instruction data in the programmed instructions.                                                   |
| 0x86  | Failure to Receive Wink                          | Layer 3 purges while waiting for wink from the distant end within the timer period.                                                                                                                                                                      | None—the CSP has no control over the remote channel.                                                                                                                            |
| 0x87  | Wink Tolerance Exceeded                          | Layer 3 received a wink that exceeded the maximum allowed wink duration. If this condition happens for the initial wink, glare is declared.                                                                                                              | If the maximum allowed wink duration is too short, increase it; otherwise, you cannot take action because the CSP does not have control over the remote channel.                |
| 0x88  | On-hook Outpulse Complete Timeout                | Layer 3 failed to get an outpulsing complete message from tone call within the Outpulse Complete Timer.<br><br>Default timer: 6 seconds.                                                                                                                 | Modify the <i>Outseize Control</i> message or programmed outseize instructions so that the correct number of digits is being outpulsed, or increase the outpulse complete time. |
| 0x89  | Failure to Detect Off-hook                       | Reserved.                                                                                                                                                                                                                                                | Contact Dialogic Tech Support                                                                                                                                                   |
| 0x8A  | Failure to Receive Call Progress Analysis Result | Layer 3 failed to get a call progress analysis result within the timer period.<br><br>Default timer: 120 seconds.                                                                                                                                        | Contact Dialogic Tech Support                                                                                                                                                   |
| 0x8B  | Failure to Detect Dialtone                       | Layer 3 failed to get dialtone detection from call progress analysis within the timer period.<br><br>Default timer: 120 seconds.                                                                                                                         | Verify outgoing start dial type.                                                                                                                                                |
| 0x8C  | Off-hook Host Control Wait Timeout               | Layer 3 purges when a timer expires in the off-hook host control wait state. The host failed to provide further action within the timer period.<br><br>Default timer: 120 seconds.                                                                       | Provide host control within timer period.                                                                                                                                       |
| 0x8D  | Off-hook Host Data Wait Timeout                  | An Outpulse Stage N Address Data instruction has been included in either an <i>Outseize Control</i> message or preprogrammed outseize instructions without corresponding Stage N Address Data within the timer period.<br><br>Default timer: 120 seconds | Include a Stage N Address Data ICB in the <i>Outseize Control</i> message or instruction data in the programmed instructions.                                                   |
| 0x8E  | Invalid Off-hook Outseize Instruction            | Layer 3 purged attempting to Scan For Wink or Scan For ANI Off-hook because the distant end was already off-hook.                                                                                                                                        | Verify the call flow.                                                                                                                                                           |

| Value | Description                               | Comments                                                                                                                                                                                                                           | Corrective User Action                                                                                                                                                                                         |
|-------|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x8F  | Off-hook Outpulse Complete Timeout        | Layer 3 failed to get an outpulsing complete message from tone control within the timer period.<br><br>Default timer: 6 seconds.                                                                                                   | Modify the <i>Outseize Control</i> message or programmed outseize instructions so that the correct number of digits is being outpulsed, or increase the outpulse complete time.                                |
| 0x90  | Invalid Inseize Instruction Encountered   | An invalid inseize instruction was encountered while processing an <i>Inseize Control</i> message or programmed inseize instructions.                                                                                              | Modify the <i>Inseize Control</i> message or programmed inseize instruction list.                                                                                                                              |
| 0x91  | Null Inseize Instruction Encountered      | A null inseize instruction was encountered while processing an <i>Inseize Control</i> message.                                                                                                                                     | Modify the <i>Inseize Control</i> message.                                                                                                                                                                     |
| 0x92  | Null Inseize Instruction List Encountered | A Null inseize instruction was encountered while processing an inseize instruction list.                                                                                                                                           | Modify the inseize instruction list with the <i>Inseize Instruction List Configure</i> message.                                                                                                                |
| 0x93  | Initialized Impulsing Stage Data          | A Receive Stage N Address Data instruction was included in either an <i>Inseize Control</i> message or programmed inseize instruction list with impulsing parameters for stage not configured.                                     | Modify the <i>Inseize Control</i> message or inseize control list so that the correct stage number is specified; or configure the impulsing parameters with the <i>Impulsing Parameters Configure</i> message. |
| 0x94  | Invalid Inseize Start Dial Signal         | Reserved.                                                                                                                                                                                                                          | Contact Dialogic Tech Support                                                                                                                                                                                  |
| 0x95  | Invalid Inseize Receive Stage Condition   | Reserved.                                                                                                                                                                                                                          | Contact Dialogic Tech Support                                                                                                                                                                                  |
| 0x96  | Invalid Inseize Receive Stage Parameters  | A Receive Stage N Address Data instruction was included in either an <i>Inseize Control</i> message or programmed inseize instruction list with impulsing parameters for stage not configured.                                     | Modify the <i>Inseize Control</i> message or inseize control list so that the correct stage number is specified; or configure the impulsing parameters with the <i>Impulsing Parameters Configure</i> message. |
| 0x97  | No Receiver Request Response              | A Receive Stage N Address Data instruction was included in either an <i>Inseize Control</i> message or programmed inseize instruction list and Layer 3 has failed to get a response from a digit receiver within the timer period. | Verify digit receiver resources.                                                                                                                                                                               |
| 0x98  | No Digits Response                        | Layer 3 sent a request for digit collection, but failed to get a response within the timer period.                                                                                                                                 | Make sure digit collection timers are set correctly ( <i>Impulsing Parameters Configure</i> message).                                                                                                          |
| 0x99  | Inseize Host Control Wait Timeout         | Layer 3 purges when a timer expires in the Inseize Host Control Wait state. The host failed to provide further action within the specified amount of time.<br><br>Default timer: 120 seconds.                                      | Provide host control.                                                                                                                                                                                          |
| 0x9A  | No Inseize ACK Outstanding                | A Send Host ACK instruction has been included in either an Inseize Control message or programmed inseize instruction list but there are no unacknowledged inseize instructions.                                                    | Make sure there is only one Send Host ACK instruction per <i>Outseize Control</i> or outseize instruction list.                                                                                                |

| Value | Description                                                                               | Comments                                                                                                                                                                                                            | Corrective User Action                                                  |
|-------|-------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| 0x9B  | Function Type Not Supported                                                               | Layer 3 requested DSP service but the function type requested is not available.                                                                                                                                     | Make sure a DSP is configured for the function type.                    |
| 0x9C  | Suspend Reject                                                                            | Reserved.                                                                                                                                                                                                           | Contact Dialogic Tech Support                                           |
| 0x9D  | Resume Reject                                                                             | Reserved.                                                                                                                                                                                                           | Contact Dialogic Tech Support                                           |
| 0x9E  | PPL Invoked For Invalid Trunk Type                                                        | Reserved.                                                                                                                                                                                                           | Contact Dialogic Tech Support                                           |
| 0x9F  | PPL Null State/Event List                                                                 | State/Event list is null.                                                                                                                                                                                           | Check PPL and state/event table list.                                   |
| 0xA0  | PPL Invalid Event for Decision State                                                      | PPL is in an internal state, waiting for some event to drive it out to a normal state. It receives an internal event while in the internal state.                                                                   | Check the PPL.                                                          |
| 0xA1  | PPL Timer Expired in Internal State                                                       | Reserved.                                                                                                                                                                                                           | Contact Dialogic Tech Support                                           |
| 0xA2  | PPL Negative Internal State Response                                                      | A negative response was received in an internal state. Layer 3 was requesting a DSP service while the service was not available or supported.                                                                       | Make sure the MFDSP is configured for the function type.                |
| 0xA3  | PPL Not Internal Wait Event                                                               | Reserved.                                                                                                                                                                                                           | Contact Dialogic Tech Support                                           |
| 0xA4  | PPL Exceeded Maximum Number of Iterations                                                 | The maximum number of PPL iterations was exceeded.                                                                                                                                                                  | Contact Dialogic Tech Support                                           |
| 0xA5  | ISDN Host Call Proceeding Wait Timeout                                                    | A time-out occurred while in the ins_pass through state.                                                                                                                                                            | Contact Dialogic Tech Support                                           |
| 0xA6  | PPL Null Primitive                                                                        | A null primitive was encountered. There is no atomic function between two states.                                                                                                                                   | Check the PPL. Make sure there are atomic functions between two states. |
| 0xAC  | Invalid ISUP Data                                                                         | An API message such as <i>Release with Data</i> (0x0036) contains invalid ICB data.                                                                                                                                 | Correct the data in the ICB.                                            |
| 0xAF  | Config Byte Max Offset Exceeded                                                           | The config byte's max offset has been exceeded.                                                                                                                                                                     | Contact Dialogic Tech Support                                           |
| 0xB1  | Timeout Waiting for Silence in Backward Reception Mode                                    | Reserved.                                                                                                                                                                                                           | Contact Dialogic Tech Support                                           |
| 0xB2  | Timeout Waiting to Receive a Forward Tone to Transmit from PPL in Backward Reception Mode | Reserved.                                                                                                                                                                                                           | Contact Dialogic Tech Support                                           |
| 0xB3  | Timeout While Waiting to Receive a Tone in Backward Reception Mode                        | Reserved.                                                                                                                                                                                                           | Contact Dialogic Tech Support                                           |
| 0xB4  | Timeout While Waiting for Silence in Forward Reception Mode                               | Timeout occurred while waiting for the incoming forward tone to terminate after a backward signal has been sent in the forward reception state machine.<br><br>Default timer: 10 seconds (R2 timer #4 divided by 2) | Determine cause of sustained signal on external termination device.     |

| Value | Description                                                                                            | Comments                                                                                                                                                                                                                                                                                                                              | Corrective User Action                                                                                    |
|-------|--------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| 0xB5  | Timeout Waiting to Receive a Forward Tone in Forward Reception Mode                                    | The termination device that initiated the call has failed to transmit a forward tone within the time allowed after transition to the wait for forward tone within the time allowed after a transitioning to the wait for forward tone state in the forward state machine.<br><br>Default timer: 10 seconds (R2 timer #4 divided by 2) | Determine cause of signal delay on external termination device.                                           |
| 0xB6  | Timeout Waiting to Receive a Backward Tone to Transmit from PPL in Forward Reception Mode              | The PPL has failed to transmit a backward signal within the timer period after transitioning to the Wait For Backward Signal To Transmit state in the forward reception state machine.<br><br>Default timer: 10 seconds (R2 timer #4 divided by 2)                                                                                    | Determine cause of failure of PPL to transmit backward signal.                                            |
| 0xB7  | Received Silence While Waiting for an Event to Transmit a Backward Tone in Backward Reception Mode     | Indicates that the termination device that initiated the call has dropped the forward signal prior to receiving a backward acknowledgment.                                                                                                                                                                                            | Determine the cause of signal drop on external termination device.                                        |
| 0xB8  | Received an Incoming Forward Tone While Waiting to Transmit a Backward Tone in Forward Reception Mode. | Indicates a violation of compelled signaling on an incoming call. The termination device is transmitting a forward signal prior to receiving a backward acknowledgment.                                                                                                                                                               | Determine the cause of failure of external termination device to wait for backward acknowledgment signal. |
| 0xCA  | DSP Release Wait Timeout                                                                               | A DSP resource timed out while attempting to release the resource                                                                                                                                                                                                                                                                     | Contact Dialogic Technical Support.                                                                       |
| 0xCB  | VDAC purges the channel because Layer 4 did not release it.                                            | VPPL L3 on the VDAC negatively acknowledges the <i>Outseize Control</i> message and times out waiting for Layer 4 to release the channel. Since Layer 4 did not release the channel, VDAC purges the channel.                                                                                                                         |                                                                                                           |
| 0xCC  | Layer 3 PPL Purge Error                                                                                | Varies by PPL protocol.                                                                                                                                                                                                                                                                                                               |                                                                                                           |
| 0xCD  | Wait for Host Control Timeout                                                                          | Varies by PPL protocol.                                                                                                                                                                                                                                                                                                               |                                                                                                           |
| 0xCF  | Forced purge for Media Recovery                                                                        |                                                                                                                                                                                                                                                                                                                                       | None                                                                                                      |
| 0xD4  | Layer 4 Connect wait timeout                                                                           | Varies by PPL protocol.                                                                                                                                                                                                                                                                                                               |                                                                                                           |
| 0xD5  | Layer 4 Clear wait timeout                                                                             | Varies by PPL protocol.                                                                                                                                                                                                                                                                                                               |                                                                                                           |
| 0xDA  | Layer 4 Busy out ACK wait timeout                                                                      | Varies by PPL protocol.                                                                                                                                                                                                                                                                                                               |                                                                                                           |
| 0xDB  | IP Resource Release Wait Timeout                                                                       | VDAC Resource Release Wait Timeout                                                                                                                                                                                                                                                                                                    | Contact Dialogic Technical Support                                                                        |

| Value | Description                                                             | Comments                                                                                                                                                                                                                                      | Corrective User Action                                                                       |
|-------|-------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| 0xDD  | SIP Internal Error                                                      | Could be caused by any of the following:<br><br>Message size exceeds the maximum IMSG size.<br><br>NPDI data size exceeds the maximum.<br><br>Outgoing SIP message size exceeds 1500 Bytes.                                                   | Check the incoming SIP message.                                                              |
| 0xDE  | VDAC rejected/timeout                                                   | The message sent to the VDAC-ONE card was not acknowledged or timed out.                                                                                                                                                                      | Check the VDAC-ONE card and router configurations.                                           |
| 0xDF  | Network error/timeout<br><br>SIP Socket Open Failure                    | The remote SIP entity is not responding to messages.<br><br>The peer entity cannot accept any additional TCP socket connections.                                                                                                              | Check if the remote side is down.<br>Check the network.                                      |
| 0xE0  | Layer 4 response timeout                                                | SIP timed out waiting for a response from Layer 4.                                                                                                                                                                                            |                                                                                              |
| 0xE1  | Invalid SDP/NPDI data                                                   | In call agent mode, the user puts invalid data (typically sdp) in the <i>Connect With Data</i> message                                                                                                                                        |                                                                                              |
| 0xE2  | Invalid coupled channel state<br><br>or<br><br>Invalid physical channel | The physical span/channel identifier provided by the host application (typically for call gent mode coupling) is in non-idle state in the CSP.<br><br>or<br><br>The host supplied physical span channel is not a valid physical VoIP channel. | Resolve glare issues between the CSP and the host channel hunting and allocation algorithms. |
| 0xF7  | Layer 4 Clear Request timeout                                           |                                                                                                                                                                                                                                               |                                                                                              |
| 0xFD  | Call Progress Analysis timeout                                          |                                                                                                                                                                                                                                               |                                                                                              |
| 0xFF  | R2 Cycle Complete timeout                                               |                                                                                                                                                                                                                                               |                                                                                              |

## DS3 Configure/Query 0x000F

---

**SwitchKit Name** DS3GenericMessage

**Type** EXS API and SwitchKit API message

**Description** **DS3 Configure/Query 0x000F**

Use this message to perform the following:

- Configure Loopback Mode
- Query Loopback Mode
- DS3 Framer Configure and Query

**NOTE:** You must take a DS3 (Entity 1) or the DS3 spans (Entity 3) out of service before you can place it into local loop back. The DS3 AIB must always be 0.

**Sent by** Host

**SwitchKit Code** **Configuration**

```
DS3GenericMessage (
 Node = integer,
 Slot = integer,
 DS3 = integer,
 Entity = integer,
 DataLength = integer,
 Data = byte array);
```

**C Structure**

```
typedef struct {
 UBYTE Slot;
 UBYTE DS3;
 UBYTE Entity;
 UBYTE DataLength;
 UBYTE Data[221];
} XL_DS3GenericMessage;
```

**C Structure Response**

```
typedef struct {
 unsigned short Status;
 UBYTE Slot;
 UBYTE DS3;
 UBYTE Entity;
 UBYTE DataLength;
 UBYTE Data[219];
} XL_DS3GenericMessageAck;
```

**C++ Class**

```

class XLC_DS3GenericMessage : public XLC_OutboundMessage
{
public:
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
 UBYTE getDS3() const;
 void setDS3(UBYTE x);
 UBYTE getEntity() const;
 void setEntity(UBYTE x);
 UBYTE getDataLength() const;
 void setDataLength(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};

```

**C++ Class Response**

```

class XLC_DS3GenericMessageAck : public
 XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
 UBYTE getDS3() const;
 void setDS3(UBYTE x);
 UBYTE getEntity() const;
 void setEntity(UBYTE x);
 UBYTE getDataLength() const;
 void setDataLength(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};

```

**EXS API Hex Format**

| MESSAGE (White) |                       | RESPONSE (Gray) |                       |
|-----------------|-----------------------|-----------------|-----------------------|
| Byte            | Field Name            | Byte            | Field Name            |
| 0               | Frame (0xFE)          | 0               | Frame(0xFE)           |
| 1, 2            | Length (0x00NN)       | 1, 2            | Length (0x00NN)       |
| 3, 4            | Message Type (0x000F) | 3, 4            | Message Type (0x000F) |
| 5               | Reserved (0x00)       | 5               | Reserved (0x00)       |
| 6               | Sequence Number       | 6               | Same Number           |
| 7               | Logical Node ID       | 7               | Logical Node ID       |

DS3 Configure/Query 0x000F

| MESSAGE (White) |                                                                                                                                                                                                           | RESPONSE (Gray) |                        |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|------------------------|
| Byte            | Field Name                                                                                                                                                                                                | Byte            | Field Name             |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs<br><hr/> Number of AEs<br><hr/> AEs<br>0x32 DS3 Offset<br><br>This value should always be 0.                                                        | 8, 9            | Status (MSB, LSB)      |
| :               | Entity<br>The operation to perform on the DS3:<br>0x01 Loop Back Configure<br>0x02 Loop Back Query<br>0x03 DS1 Loop Back Configure<br>(All spans)<br>0x04 DS3 Frammer Configure<br>0x05 DS3 Frammer Query | 10              | AIB<br>Same as message |

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | RESPONSE (Gray) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Byte            | Field Name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Byte            | Field Name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| :               | Data Length                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | :               | Entity                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | :               | Data Length: Variable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| :               | <p>Data[0-N]<br/>This field is dependent on the entity specified:</p> <p>Entity: 0x01, 0x03:<br/>Data[0] Loop Back Mode<br/>0x00 Disable Local Loop Back<br/>Disable loop back of received stream to the transmit DS3 stream</p> <p>0x01 Enable Local Loop Back<br/>Enable loop back of received stream to the transmit stream</p> <p>0x02 Cancel Remote Loop Back<br/>Cancel the request for the distant end to loop back its received stream to the transmit stream (CBIT only)</p> <p>0x03 Request Remote Loop Back<br/>Request distant end to loop back its received stream to the transmit stream (CBIT only)</p> <p>Entity: 0x02<br/>Data: No data required</p> <p>Entity: 0x04 - DS3 Framer Configure<br/>Data [0]<br/>0x00 - CBIT<br/>0x01 - M13</p> <p>Entity: 0x05 DS3 Framer Query<br/>Data: No data required</p> | :               | <p>Info[0-N]<br/>Information returned depends upon the entity specified:</p> <p>Entity: 0x01, 0x03:<br/>No information reported.</p> <p>Entity: 0x02:<br/>Info[0] Local Loop Back Mode<br/>0x00 Local Loop Back Disabled<br/>0x01 Local Loop Back Enabled</p> <p>Info[1] Remote Loop Back Mode<br/>0x00 Remote Loop Back Requested<br/>0x01 Remote Loop Back Cancelled</p> <p>When the distant end requests that the CSP be placed into loopback mode, an alarm is sent to the host. The host is responsible for initiating the loopback process.</p> <p>Entity: 0x04 - DS3 Framer Configure<br/>No information reported.</p> <p>Entity: 0x05 DS3 Framer Query<br/>Data[0]<br/>0x00 - CBIT<br/>0x01 - M13</p> |
| :               | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | :               | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

# DSP Cache Modify 0x011A

---

**SwitchKit Name** DSPCacheModify

**Type** EXS API and SwitchKit API message

**Description** **DSP Cache Modify 0x011A**

**NOTE:** This message applies to the DSP Series 2 card only.

Use this message to clear either a single file ID (including a file ID for a temporary recording) or all file IDs on the main board cache as well as the cache on all the DSPs. You can also use this message to copy any file residing in cache (playback or recording) to permanent storage on a server.

**Sent by** Host Application

**SwitchKit Code** **C Structure**

```
typedef struct {
 UBYTE Slot;
 UBYTE reserved18[29];
 UBYTE DataType;
 UBYTE TLVCount;
 UBYTE Data[220];
} XL_DSPCacheModify;
```

**C++ Class**

```
class XLC_DSPCacheModify : public XLC_OutboundMessage {
public:
 UBYTE getSlot() const ;
 void setSlot(UBYTE x);
 UBYTE getDataType() const;
 void setDataType(UBYTE x);
 UBYTE getTLVCount() const;
 void setTLVCount(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                       | RESPONSE (Gray) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------|-------------------------------------------------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Byte            | Field Description                                     | Byte            | Field Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 0               | Frame (0xFE)                                          | 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 1, 2            | Length (0x00NN)                                       | 1, 2            | Length (0x00NN)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 3, 4            | Message Type (0x011A)                                 | 3, 4            | Message Type (0x011A)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 5               | Reserved (0x00)                                       | 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 6               | Sequence Number                                       | 6               | Same Sequence Number                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 7               | Logical Node ID                                       | 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs | 8, 9            | Status (MSB, LSB)<br>0x0001 Invalid TLV Data – Possible reasons for this Nack are as follows:<br><br>- File ID is not cached.<br>- File cannot be written to the NFS server. This could be due to multiple reasons.<br><br>- File not present in cache<br>- NFS server not mounted<br>- NFS server is mounted but the file could not be opened/written (for example, if the share does not have write permission or the server runs out of space)<br><br>0x0003 Invalid number of TLVs<br>There are no TLVs in the message.<br>0x0004 Invalid TLV Length<br>The TLV length is different from what is expected.<br>0x0006 Invalid TLV<br>Unknown TLV.<br>0x000D Mandatory TLVs missing<br>One or more mandatory TLVs is missing.<br><br>Also see Common Response Status Values in the <i>API Reference</i> |
|                 | Number of AEs to follow                               | 10              | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|                 | AE<br>0x01 Slot                                       |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| :               | Data Type<br>0x00 TLVs                                |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

## DSP Cache Modify 0x011A

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | Number of TLVs to Follow                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| : | <p>TLVs</p> <p>0x05FA Card Object</p> <p>0x05FF DSP Cache Modify Action</p> <p>If the DSP Cache Modify Action TLV specifies Clear Specific Cache (0x0000), then one or more File ID TLVs (0x05E0) are required.</p> <p>If the DSP Cache Modify Action TLV specifies Copy Specific Cache (0x0002), then the File ID (0x05E0), File Format (0x05E1) and File Location (0x05E2) TLVs are required</p> <p>If the DSP Cache Modify Action TLV specifies Clear All (0x0001), then no other TLVs are required besides 0x05FA Card Object</p> <p>0x05E0 File ID</p> <p>0x05E1 File Format</p> <p>0x05E2 File Location</p> |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

# DSP Service Cancel 0x00BE

---

**SwitchKit Name** DSPServiceCancel

**Type** EXS API and SwitchKit API message

**Description** **DSP Service Cancel 0x00BE**  
Use this message to cancel a DSP Service.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
 UBYTE ServiceType;
 UBYTE reserved48[222];
} XL_DSPServiceCancel;
```

**C++ Class**

```
class XLC_DSPServiceCancel : public
 XLC_OneChannelOutbound {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 UBYTE getServiceType() const;
 void setServiceType(UBYTE x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                       | RESPONSE (Gray) |                       |
|-----------------|-----------------------|-----------------|-----------------------|
| Byte            | Field Description     | Byte            | Field Description     |
| 0               | Frame (0xFE)          | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)       | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x00BE) | 3, 4            | Message Type (0x00BE) |
| 5               | Reserved (0x00)       | 5               | Reserved (0x00)       |
| 6               | Sequence Number       | 6               | Same Sequence Number  |
| 7               | Logical Node ID       | 7               | Logical Node ID       |

DSP Service Cancel 0x00BE

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |      |                 |
|---|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-----------------|
| : | <u>AIB</u>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 8, 9 | Status MSB, LSB |
|   | Address Method<br>0x00 - Individual AEs                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 10   | Checksum        |
|   | Number of AEs                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |      |                 |
|   | AE<br>0x0D Channel                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |      |                 |
|   | Service Type<br>0x01 DTMF Receiver<br>0x02 Energy Detection<br>0x03 CPA Receiver<br>0x04 Coin Tone Receiver<br>0x14 DTMF and Dial Pulse Receiver<br>0x15 Dial Pulse Receiver<br>0x16 DTMF and Dial Pulse Receiver with Termination of Tone/Announcement<br>0x17 Dial Pulse Receiver with Termination of Tone/Announcement<br>0x81 DTMF Receiver, Matrix (Based RFC 2833 Collection)<br>0xC1 DTMF Receiver, Matrix (Based on SUBSRIBE/NOTIFY)<br>0xE1 DTMF Receiver with Termination of Tone/Announcement |      |                 |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |      |                 |

# DSP Service Request 0x00BD

---

**SwitchKit Name** DSPServiceRequest

**Type** EXS API and SwitchKit API message

**Description** **DSP Service Request 0x00BD**

Use this message to allocate a DSP service on the specified channel. DSP services supported are:

- DTMF Reception
- Energy Detection
- Call Progress Analysis Reception
- Coin Tone Reception (DSP-ONE Only)

After this message is sent the CSP sends the *Call Processing Event* (0x2E) message to report the occurrence of a corresponding Event on the specified channel. However, if the DSP service type is *Call Progress Analysis* (0x03) the CSP answers with a *Call Progress Analysis Result* (0x34) message.

**NOTE:** Ringback is detected on the fourth cycle of ringback/silence.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
 UBYTE ServiceType;
 UBYTE Data[222];
} XL_DSPServiceRequest;
```

**C++ Class**

```
class XLC_DSPServiceRequest : public
 XLC_OneChannelOutbound {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 UBYTE getServiceType() const;
 void setServiceType(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | RESPONSE (Gray) |                                                          |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|----------------------------------------------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Byte            | Field Description                                        |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 0               | Frame (0xFE)                                             |
| 1, 2            | Length (0x00NN)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 1, 2            | Length (0x0007)                                          |
| 3, 4            | Message Type (0x00BD)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 3, 4            | Message Type (0x00BD)                                    |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 5               | Reserved (0x00)                                          |
| 6               | Sequence Number                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 6               | Same Sequence Number                                     |
| 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 7               | Logical Node ID                                          |
| :               | AIB<br>Address Method<br>0x00 - Individual AEs                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 8, 9            | Status (MSB, LSB)<br>0x0002 DSP Receiver Request Timeout |
|                 | Number of AEs to follow                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 10              | Checksum                                                 |
|                 | AE<br>0x0D Channel                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                 |                                                          |
| :               | Service Type<br>0x01 DTMF Receiver<br>0x02 Energy Detection<br>0x03 Call Progress Analysis Receiver<br>0x04 Coin Tone Receiver<br>0x14 DTMF and Dial Pulse Receiver<br>0x15 Dial Pulse Receiver<br>0x16 DTMF and Dial Pulse Receiver with Termination of Tone/Announcement<br>0x17 Dial Pulse Receiver with Termination of Tone/Announcement<br>0x81 DTMF Receiver, Matrix Based RFC 2833 collection<br>0xC1 DTMF Receiver, Matrix Based on SUBSCRIBE/NOTIFY<br>0xE1 DTMF Receiver with Termination of Tone/Announcement |                 |                                                          |

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | <p>Data[0 . . . ]</p> <p>The meaning of the <i>Data</i> fields depends on the <i>Service Type</i>. See below.</p> <p>0x01 DTMF Receiver (0x01) and<br/>         0x16 DTMF and Dial Pulse Receiver with Termination of Tone/Announcement<br/>         0x17 Dial Pulse Receiver with Termination of Tone/Announcement<br/>         0x81 DTMF Receiver, Matrix Based RFC 2833 Collection<br/>         0xE1 DTMF Receiver with Termination of Tone/Announcement</p> <p>The DTMF receiver data instructs the CSP to attach a digit receiver to the specified channel and to start collecting digits. The call must be in a state other than idle, out of service, or busied out. The CSP sends Call Processing Event messages with event "digits" for every digit collected. If an error condition occurs, the CSP sends a Call Processing Event message with the "Status" field indicating the error.</p> <p>The host application is responsible for instructing the CSP to release the digit receiver when either the receiver is finished collecting digits or when an error condition occurs. The host accomplishes the release by sending a DSP Service Cancel message. If the call is disconnected, the CSP releases the digit receiver.</p> <p>Data[0,1] Digit Timer<br/>         The maximum amount of time to wait for the digit.<br/>         The following values disable this timer:<br/>         0xFFFF<br/>         0x0000<br/>         0x0001<br/>         0x0002</p> <p>Data[2] Configuration Bits (0 = disabled, 1 = enabled)<br/>         Bit 0 Report the digit at its falling edge (when the first digit is released).<br/>         If the bit is not set the digit will be reported at its rising edge (when the digit is first entered).<br/>         Bit 1 Ignore "#" character as first digit.<br/>         Bit 2 Report digit duration (the amount of time the digit has been pressed).<br/>         Valid only if Configuration Bit Number 0 is 1.<br/>         Bits 3–7 Reserved, must be 0.</p> <p>Data[3,4] Minimum Receive Digit Duration Timer<br/>         Valid timer values: 0x0000–0xFFFE</p> <p>The minimum amount of time for a digit to be present before it is detected and validated.<br/>         If you send the values 0x0000 through 0x0002, the CSP automatically sets the time to 0x0003.</p> <p>Note: The Minimum Receive Digit Duration Timer field is not valid for dial pulse detection.<br/>         For DTMF detection, you may set either the Digit Timer or the Minimum Digit Duration timer to 0xFFFF, but <b>not both</b>. If you set both timers to 0xFFFF, this message cannot detect digits.</p> <p>Data[5] Termination Digit for Playback/Tone Generation<br/>         0xFF Any DTMF termination digit for terminating playback/tone generation</p> <p>Note: Data[5] used with <i>Service Type</i> 0xE1, 0x16 and 0x17</p> |
|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x02 | <p>Energy Detection</p> <p>Data[0] Sensitivity Level<br/>                 The loudness is measured by the energy level present for white noise (constant energy) at a specified dBm level, ranging from 0 dBm (loudest) to -30 dBm (quietest). A sensitivity level of 0 dBm, the least sensitive, would be used in a case where there is a high level of background noise expected. A sensitivity level of -30 dBm, the most sensitive would be used if there is little background noise expected.</p> <p>0x00 0 dBm<br/>                 0x01 05 dBm<br/>                 0x02 10 dBm<br/>                 0x03 15 dBm<br/>                 0x04 20 dBm<br/>                 0x05 25 dBm<br/>                 0x06 30 dBm</p> <p>Data[1] Reporting Mode<br/>                 0x01 Report Initial Energy Detection Only<br/>                 0x02 Report All Energy Threshold Crossings</p> <p>Data[2,3] Scan Duration<br/>                 The scan duration must be expressed in 20 ms increments (10 ms units). For example:<br/>                 0x0002 = 20ms<br/>                 0x0004 = 40 ms<br/>                 0x0006 = 60 ms, etc.</p> <p>Odd increments are rounded down (for example, 0x0003 = 20ms). For more information on energy detection, refer to the <i>API Developer's Guide: Overview</i>.</p> <p>Data[4,5] Completion Timer<br/>                 The maximum amount of time to scan for energy. The following values disable this timer:<br/>                 0xFFFF<br/>                 0x0000<br/>                 0x0001<br/>                 0x0002</p> |
| 0x03 | <p>Call Progress Analysis Receiver<br/>                 Allows the host to perform interactive CPA on a channel.</p> <p>Data[0] CPA Class<br/>                 0x00 North American Default<br/>                 0x01 Dialtone<br/>                 0x02 CPC Detection<br/>                 0x03 Energy Detection</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 0x04 | <p>Coin Tone Receiver<br/>                 Data[0,1] Completion Timer<br/>                 The maximum amount of time to scan for Coin Tones. If the completion timer expires, the CSP releases the Coin Tone receiver. This field may be set to 0xFFFF to disable the timer.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| :    | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

# DSP SIMM Configure 0x00C0

---

**SwitchKit Name** DSPSIMMConfig

**Type** EXS API and SwitchKit API message

**Description** **DSP SIMM Configure 0x00C0**

Use this message to configure a SIMM on a DSP card. Each SIMM is populated with four DSPs. The host can specify the function to be downloaded to each DSP.

You must take a SIMM out of service before sending this message. After receiving an Acknowledgment to the *DSP SIMM Configure* message, bring the SIMM back in service.

On the DSP-ONE card, each DSP can only handle one function, and four DSPs are configured at the same time.

On the DSP Series 2 card, each DSP can handle multiple functions, so the DSP Chip AIB (0x22) is used to configure individual DSP chips.

**NOTE:** When configuring DSP cards for MFR2 signaling, each card that contains MFR2 receivers must also have MFR2 backward and forward transmitters. If MFR2 Receivers only are present, an incoming call causes an alarm that specifies “DSP Function Not Configured” and the MFR2 receivers are not used.

**Sent by** Host Application

**Example Message (Socket Log Output for SwitchKit)** 00 0F 00 C0 00 00 FF 00 01 12 02 01 01 01 01 01

**SwitchKit Code** **Configuration**

```
DSPSIMMConfig (
 Node = integer,
 Slot = integer,
 SIMM = integer,
 DSPChip = integer,
 DSP0Func = integer,
 DSP1Func = integer,
 DSP2Func = integer,
 DSP3Func = integer);
```

**C Structure**

```
typedef struct {
 UBYTE AddrInfo[30];
 UBYTE DSP0Func;
 UBYTE DSP1Func;
 UBYTE DSP2Func;
 UBYTE DSP3Func;
```

```
} XL_DSPSIMMConfig;
```

**C++ Class**

```
class XLC_DSPSIMMConfig : public XLC_SlotMessage {
public:
 const UBYTE *getAddrInfo() const;
 UBYTE *getAddrInfo();
 void setAddrInfo(UBYTE *x);
 UBYTE getDSPChip() const;
 void DSPChip(UBYTE x);
 UBYTE getDSPSlot() const;
 void setDSPSlot(UBYTE x);
 UBYTE getDSPSIMM() const;
 void setDSPSIMM(UBYTE x);
 UBYTE getSIMM() const;
 void setSIMM(UBYTE x);
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
 UBYTE getDSP0Func() const;
 void setDSP0Func(UBYTE x);
 UBYTE getDSP1Func() const;
 void setDSP1Func(UBYTE x);
 UBYTE getDSP2Func() const;
 void setDSP2Func(UBYTE x);
 UBYTE getDSP3Func() const;
 void setDSP3Func(UBYTE x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                    | RESPONSE (Gray) |                       |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                  | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                       | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)                                                                                                                                    | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x00C0)                                                                                                                              | 3, 4            | Message Type (0x00C0) |
| 5               | Reserved (0x00)                                                                                                                                    | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                    | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                    | 7               | Logical Node ID       |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs<br>Number of AEs to follow<br>AEs<br>0x12 DSP SIMM<br>0x22 DSP Chip (DSP Series 2 card only) | 8, 9            | Status MSB, LSB       |
| :               | Function 1                                                                                                                                         | 10              | Checksum              |
| :               | Function 2                                                                                                                                         |                 |                       |
| :               | Function 3                                                                                                                                         |                 |                       |
| :               | Function 4                                                                                                                                         |                 |                       |
| :               | Checksum                                                                                                                                           |                 |                       |

**Slot Number** The slot numbers are as follows:  
CSP 2090: 0x00–0x0F - Physical slot number of the DSP card  
CSP 2040: 0x00–0x03 - Physical slot number of the DSP card

**SIMM Number** The SIMM numbers are as follows:  
DSP-ONE Card: 0x00–0x03  
DSP Series 2 Card: 0x00 or 0x01

**Function** On the DSP Series 2 card, each field configures a function on a single chip on a DSP. On the DSP-ONE card, each field configures a function on an entire DSP.

#### **DSP Series 2**

Function 1 = RCV 0  
Function 2 = TXMT 0  
Function 3 = RCV 1  
Function 4 = TXMT 1

#### **DSP-ONE**

Function 1 = DSP 0  
Function 2 = DSP 1  
Function 3 = DSP 2  
Function 4 = DSP 3

**Function Types** 0x00 - No Function Type

#### **Tone Reception**

0x01- DTMF  $\mu$ -law  
0x02 - MFR1  $\mu$ -law  
0x03 - DTMF A-Law  
0x04 - MFR1 A-Law  
0x05 - MFR2 A-Law  
0x06 - MFR2  $\mu$ -Law  
0x07 - CPA A-Law  
0x08 - CPA  $\mu$ -Law  
0x09 - E1 Dial Pulse (DSP-ONE only)  
0x0A - Energy Detection  
0x0E - Coin Detection  $\mu$ -Law (DSP-ONE only)  
0x0F - Coin Detection A-Law (DSP-ONE only)

## **Tone Generation**

### DSP Series 2

0x30 - Universal Tone Generation  $\mu$ -Law

0x31 - Universal Tone Generation A-Law

### DSP-ONE

0x10 - DTMF  $\mu$ -Law

0x11 - DTMF A-Law

0x12 - MFR1  $\mu$ -Law

0x13 - MFR1 A-Law

0x14 - CPT4  $\mu$ -Law

0x15 - CPT4 A-Law

0x16 - MFR2 Forward A-Law

0x17 - MFR2 Forward  $\mu$ -Law

0x18 - MFR2 Backward A-Law

0x19 - MFR2 Backward  $\mu$ -Law

0x1A - Bong Tone  $\mu$ -Law

0x1B - Bong Tone A-Law

## **Fax**

0x32 - Send/Receive Fax

## **File Playback / Record (DSP Series 2 Only)**

0x1D - File Playback / Record

## **Conferencing**

### DSP Series 2

0x21 - Monitor

0x22 - Unified

0x23 - DTMF-Clamped \*

\* This option is for backward compatibility. With the DSP Series 2 card you can create a Unified Conference with DTMF Clamping enabled, which is the recommended method

### DSP-ONE

0x1E - Standard  $\mu$ -Law

0x1F - Standard A-Law

0x20 - Mixed

0x21 - Monitor

0x22 - Unified

0x23 - DTMF-Clamped

NOTE: You cannot configure a VRAS SIMM.

**Miscellaneous**

0x33 - Echo Cancel (DSP Series 2 Only)

0x34 - Positive Voice Detection/Answering Machine Detection (PVD/AMD) (DSP Series 2 Only)

**Frequency Shift Keying**

0x35 - Transmit FSK only (DSP Series 2 Only)

0x36 - Transmit and Receive FSK (DSP Series 2 Only)

**Default Function Types on the DSP Series 2 card**

| Module Number | DSP Number | Rcv 0                       | Txmt 0                      | Rcv 1                       | Txmt 1                      |
|---------------|------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| 0             | 0          | DTMF $\mu$ -law (0x01)      | Universal $\mu$ -law (0x30) | DTMF $\mu$ -law (0x01)      | Universal $\mu$ -law (0x30) |
| 0             | 1          | DTMF $\mu$ -law (0x01)      | Universal $\mu$ -law (0x30) | DTMF $\mu$ -law (0x01)      | Universal $\mu$ -law (0x30) |
| 0             | 2          | CPA $\mu$ -law (0x08)       | Universal $\mu$ -law (0x30) | CPA $\mu$ -law (0x08)       | Universal $\mu$ -law (0x30) |
| 0             | 3          | File Playback/Record (0x1D) | Not allowed (0x00)          | File Playback/Record (0x1D) | Not allowed (0x00)          |
| 1             | 0          | DTMF $\mu$ -law (0x01)      | Universal $\mu$ -law (0x30) | DTMF $\mu$ -law (0x01)      | Universal $\mu$ -law (0x30) |
| 1             | 1          | DTMF $\mu$ -law (0x01)      | Universal $\mu$ -law (0x30) | DTMF $\mu$ -law (0x01)      | Universal $\mu$ -law (0x30) |
| 1             | 2          | MFR1 $\mu$ -law (0x02)      | Universal $\mu$ -law (0x30) | MFR1 $\mu$ -law (0x02)      | Universal $\mu$ -law (0x30) |
| 1             | 3          | MFR1 $\mu$ -law (0x02)      | Universal $\mu$ -law (0x30) | MFR1 $\mu$ -law (0x02)      | Universal $\mu$ -law (0x30) |

**Configuration Options on the DSP Series 2 Card**

The following table shows configuration options for function types on the DSP Series 2 card.

| Rcv 0                               | Txmt 0                   | Rcv 1                               | Txmt 1                   |
|-------------------------------------|--------------------------|-------------------------------------|--------------------------|
| Any Tone Receiver Function          | Universal Tone Generator | Any Tone Receiver Function          | Universal Tone Generator |
| File Record or Playback             | Not allowed (0x00)       | File Record or Playback             | Not allowed (0x00)       |
| T.30 Fax                            | Not allowed (0x00)       | T.30 Fax                            | Not allowed (0x00)       |
| Echo Cancel                         | Not allowed (0x00)       | Echo Cancel                         | Not allowed (0x00)       |
| Any of the Valid Conferencing Types | Not allowed (0x00)       | Any of the Valid Conferencing Types | Not allowed (0x00)       |

**Configuration FSK Options on the DSP Series 2 Card**

The following table shows configuration options for the FSK function types on the DSP Series 2 card.

| Rcv 0                           | Txmt 0                   | Rcv 1                           | Txmt 1                   |
|---------------------------------|--------------------------|---------------------------------|--------------------------|
| Any Tone Receiver Function      | Transmit FSK Only (0x35) | Any Tone Receiver Function      | Transmit FSK Only (0x35) |
| Transmit and Receive FSK (0x36) | Not allowed (0x00)       | Transmit and Receive FSK (0x36) | Not allowed (0x00)       |

**Default Function Types on the DSP-ONE Card**

For more information, please refer to the DSP Resources of the *Developer's Guide: Overview*.

The table below shows the default SIMM and DSP configurations.

|        | DSP 0                     | DSP 1                     | DSP 2                     | DSP 3                     |
|--------|---------------------------|---------------------------|---------------------------|---------------------------|
| SIMM 0 | 0x01<br>μ-law<br>DTMF Rec | 0x01<br>μ-law<br>DTMF Rec | 0x01<br>μ-law<br>DTMF Rec | 0x01<br>μ-law<br>DTMF Rec |
| SIMM 1 | 0x10<br>μ-law<br>DTMF Gen | 0x12<br>μ-law<br>MFR1 Gen | 0x14<br>μ-law<br>CPT4 Gen | 0x08<br>μ-law<br>CPA Rec  |
| SIMM 2 | 0x02<br>μ-law<br>MFR1 Rec | 0x02<br>μ-law<br>MFR1 Rec | 0x02<br>μ-law<br>MFR1 Rec | 0x02<br>μ-law<br>MFR1 Rec |
| SIMM 3 | 0x01<br>μ-law<br>DTMF Rec | 0x01<br>μ-law<br>DTMF Rec | 0x01<br>μ-law<br>DTMF Rec | 0x01<br>μ-law<br>DTMF Rec |

# DynamicConfig

---

|                    |                                                                                                                                                                                                                                                                                         |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Type</b>        | EXS SwitchKit API message                                                                                                                                                                                                                                                               |
| <b>Purpose</b>     | Use the <i>SK_DynamicConfig</i> message to dynamically update the CSP configuration without restarting the SwitchManager. This message includes an argument, ConfigMode, which determines whether the specified file content will be saved by SwitchManager for future reconfiguration. |
| <b>Description</b> | When SwitchManager receives a DynamicConfig message, it will send the configuration referenced in the message to the CSP immediately. Those messages are maintained on the CSP. The filename must be relative to the SwitchManager process.                                             |
| <b>Sent by</b>     | Converged Services Administrator or Application                                                                                                                                                                                                                                         |
| <b>Arguments</b>   | The following table shows the user modifiable arguments for this message:                                                                                                                                                                                                               |

| Argument   | Description                                                                                                                                                                                                                                                                                                                                       |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ConfigMode | This enables or disables writing the content of the specified file into the current configuration file used by Switch Manager. The options are:<br><br>SK_CONFIGSWITCH_APPEND - 0x00<br>SK_CONFIGSWITCH_REPLACE - 0x01<br>SK_CONFIGSWITCH_REPLACE_AND_SEND - 0x02<br>SK_CONFIGSWITCH_APPEND_NOSEND - 0x03<br>SK_CONFIGSWITCH_SEND_NOAPPEND - 0x04 |
| Filename   | The filename of the file containing the configuration information to parse and send. Location should be relative to SwitchManagers installation directory.                                                                                                                                                                                        |

**C Structure**

```
typedef struct {
 UBYTE ConfigMode;
 char Filename[230];
} SK_DynamicConfig;
```

**C Structure Response**

```
typedef struct {
 int Status;
} SK_DynamicConfigAck;
```

**C++ Class**

```
class SKC_DynamicConfig : public SKC_ToolkitMessage {
```

```
public:
 UBYTE getConfigMode() const;
 void setConfigMode(UBYTE x);
 const char *getFilename() const;
 void setFilename(const char *x);
};
```

**C++ Class Response**

```
class SKC_DynamicConfigAck : public SKC_ToolkitAck {
public:
 int getStatus() const;
 void setStatus(int x);
 const UBYTE *getReserved() const;
 UBYTE *getReserved();
 void setReserved(UBYTE *x);
};
```

## E1 Span Configure 0x00D8

---

**SwitchKit Name** E1SpanConfig

**Type** EXS API and SwitchKit API message

**Description** **E1 Span Configure 0x00D8**

This message allows the host to configure the transmission format, line length, and SA bits of an E1 span.

**NOTE:** Before changing span configuration, Dialogic ALWAYS RECOMMENDS that you de-assign the spans, assign them again and then send the new span configuration.

**Sent by** Host

**Example Message**

The following sample message configures an E1 span with the default coding method, signaling method, E1 Layer I management. CRC-4 and FEBE error checking are configured. The default line length, SA Registers, and TS 16 Register are also configured.

```
00 0F 00 D8 00 FF FF 00 01 0C 02 01 23 E8 00 F8 D0
```

**SwitchKit Code** **Configuration**

```
E1SpanConfig (
 Node = integer,
 Span = integer,
 Format = integer,
 LineLength = integer,
 SARegisters = integer,
 TS16Registers = integer);
```

**C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Format;
 UBYTE LineLength;
 UBYTE SARegisters;
 UBYTE TS16Registers;
} XL_E1SpanConfig;
```

**C++ Class**

```
class XLC_E1SpanConfig : public XLC_SpanMessage {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getFormat() const;
 void setFormat(UBYTE x);
 UBYTE getLineLength() const;
```

```

void setLineLength(UBYTE x);
UBYTE getSARegisters() const;
void setSARegisters(UBYTE x);
UBYTE getTS16Registers() const;
void setTS16Registers(UBYTE x);
};

```

**EXS API Hex Format**

| MESSAGE (White) |                                         | RESPONSE (Gray) |                       |
|-----------------|-----------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                       | Byte            | Field Description     |
| 0               | Frame (0xFE)                            | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)                         | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x00D8)                   | 3, 4            | Message Type (0x00D8) |
| 5               | Reserved (0x00)                         | 5               | Reserved (0x00)       |
| 6               | Sequence Number                         | 6               | Same Sequence Number  |
| 7               | Logical Node ID                         | 7               | Logical Node ID       |
| :               | <u>AIB</u>                              | 8, 9            | Status MSB, LSB       |
|                 | Address Method<br>0x00 - Individual AEs | 10              | Checksum              |
|                 | Number of AEs to follow                 |                 |                       |
|                 | AE<br>0x0C Logical Span                 |                 |                       |

E1 Span Configure 0x00D8

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | <p><b>Format</b><br/> This field is a bit mask. The meaning of each bit is described below.<br/> Unless otherwise stated, 0 = Disabled, 1 = Enabled.</p> <p>0x01 - default for the CSP</p> <p><b>Coding Method</b><br/> Bit 0<br/> 0 Alternate Mark Inversion (AMI)<br/> 1 High-Density Bipolar Three-Bit Substitution (HDB3) (Default)</p> <p><b>Error Checking</b><br/> Bit 1 Enable CRC-4 Cyclic Redundancy Check</p> <p>Bit 2 Enable FEBE (Far-End Block Error)<br/> Enables the use of E-bits in the CRC-4 multiframe alignment signal for the automatic reporting of a CRC error received during the previous multiframe.</p> <p><b>Signaling Method</b><br/> Bit 3<br/> 0 Channel Associated Signaling (Default)<br/> In CAS mode, in-band line signaling for voice channels is carried in timeslot 16.<br/> 1 Clear Channel<br/> In Clear Channel mode, in-band line signaling is not transmitted in timeslot 16.</p> <p>Bit 4 Transmit all zeroes</p> <p>Bit 5<br/> 0 E1 Layer I management (Default)<br/> 1 Euro-ISDN E1 Layer I management<br/> For Euro-ISDN compliant E1, see Notes 4 and 5.</p> <p>Bit 6<br/> 0 E1 Layer I management (Default)<br/> 1 Austel-ISDN E1 Layer I management<br/> For Austel-ISDN compliant E1, see Notes 4 and 6.</p> <p>Bit 7 Reserved, must be 0.</p> |
| : | <p><b>Line Length</b><br/> 0x00 G.703 ITU-T (75 ohm) (Default)<br/> 0x04 G.703 ITU-T (120 ohm)</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | <p><b>SA Registers</b></p> <p>This field is a bit mask. The meaning of each bit is described below.<br/>Unless otherwise stated, 0 = Disabled, 1 = Enabled. The default for all bits is 1. See Note 3.</p> <p>Bits 0–2 Reserved (must be 0)</p> <p>Bit 3 Value to which all bits in SA Bit 4 Register should be set.</p> <p>Bit 4 Value to which all bits in SA Bit 5 Register should be set.</p> <p>Bit 5 Value to which all bits in SA Bit 6 Register should be set.</p> <p>Bit 6 Value to which all bits in SA Bit 7 Register should be set.</p> <p>Bit 7 Value to which all bits in SA Bit 8 Register should be set.</p> |
| : | <p><b>TS 16 Register</b></p> <p>This field is a bit mask. The meaning of each bit is described below.<br/>Unless otherwise stated, 0 = Disabled, 1 = Enabled. The default for all bits is 1. See Note 3.</p> <p>Bits 0–3 Reserved (must be 0)</p> <p>Bit 4 Value to which spare bit x1 in frame 0 of timeslot 16 should be set.</p> <p>Bit 5 Reserved (must be 0)</p> <p>Bit 6 Value to which spare bit x3 in frame 0 of timeslot 16 should be set.</p> <p>Bit 7 Value to which spare bit x4 in frame 0 of timeslot 16 should be set.</p>                                                                                    |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

## NOTES

1. To configure a single node for greater than 64 spans, please contact Dialogic technical support.
2. Note that the Dialogic message bit-numbering format is 0–7, while the SA Register and TS 16 Register formats are 1–8. Therefore, bit 3 of the host message format is bit 4 of the SA Register and TS 16 Register formats, and so on.
3. This message is backward-compatible to previous software releases.
4. Refer to ITU-T Recommendation G.704 more details on programming SA bits and spare timeslot 16 bits.
5. To use Euro-ISDN Layer 1, you must use an SE1LC Rev. B11 or greater. For more information, refer to the appropriate *“Hardware Product Descriptions.”*
6. For Euro-ISDN compliant E1, set the following:
  - (Bit 0 = 1) Line format to HDB3
  - (Bit 1 = 1) CRC
  - (Bit 2 = 1) FEBE
  - (Bit 3 = 1) E1 clear Channel
  - (Bit 5 = 1) Euro-ISDN Layer I
  - Line length to G.703
7. For Austel-ISDN compliant E1, set the following:
  - (Bit 0 = 1) Line format to HDB3
  - (Bit 1 = 1) CRC
  - (Bit 2 = 1) FEBE
  - (Bit 3 = 1) E1 clear Channel
  - (Bit 6 = 1) Austel-ISDN Layer I
  - Line length to G.703

**Default E1 Format**

The table below shows the default format for E1 spans. You must enable Clear Channel signaling to use timeslot 16 (Channel 30) as a voice channel or for out-of-band signaling (for example, an ISDN application).

**NOTE:** Clear Channel signaling must be enabled to use timeslot 16 (channel 30) as a D channel in ISDN applications.

| Timeslot | Channel | Description                  |
|----------|---------|------------------------------|
| 0        | 31      | Alarms and framing           |
| 1        | 0       | Voice channels               |
| 2        | 1       |                              |
| 3        | 2       |                              |
| 4        | 3       |                              |
| 5        | 4       |                              |
| 6        | 5       |                              |
| 7        | 6       |                              |
| 8        | 7       |                              |
| 9        | 8       |                              |
| 10       | 9       |                              |
| 11       | 10      |                              |
| 12       | 11      |                              |
| 13       | 12      |                              |
| 14       | 13      |                              |
| 15       | 14      |                              |
| 16       | 30      | In-band line signaling (CAS) |
| 17       | 15      | Voice channels               |
| 18       | 16      |                              |
| 19       | 17      |                              |
| 20       | 18      |                              |
| 21       | 19      |                              |
| 22       | 20      |                              |
| 23       | 21      |                              |
| 24       | 22      |                              |
| 25       | 23      |                              |
| 26       | 24      |                              |
| 27       | 25      |                              |
| 28       | 26      |                              |
| 29       | 27      |                              |
| 30       | 28      |                              |
| 31       | 29      |                              |

## E1 Span Query 0x00DB

---

**SwitchKit Name** E1SpanFormatQuery

**Type** EXS API and SwitchKit API message

**Description** **E1 Span Query 0x00DB**

Use this message to retrieve the transmission format and line length of an E1 span.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short Span;
} XL_E1SpanFormatQuery;
```

### C Structure Response

```
typedef struct {
 unsigned short Status;
 unsigned short Span;
 UBYTE Format;
 UBYTE LineLength;
 UBYTE Slot;
 UBYTE Offset;
 UBYTE SRegisters;
 UBYTE TS16Register;
} XL_E1SpanFormatQueryAck;
```

### C++ Class

```
class XLC_E1SpanFormatQuery : public XLC_OutboundMessage
{
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
};
```

### C++ Class Response

```
class XLC_E1SpanFormatQueryAck : public
 XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getFormat() const;
 void setFormat(UBYTE x);
 UBYTE getLineLength() const;
 void setLineLength(UBYTE x);
```

```

UBYTE getSlot() const;
void setSlot(UBYTE x);
UBYTE getOffset() const;
void setOffset(UBYTE x);
UBYTE getSARegisters() const;
void setSARegisters(UBYTE x);
UBYTE getTS16Register() const;
void setTS16Register(UBYTE x);
};

```

**EXS API Hex Format**

| MESSAGE (White)           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | RESPONSE (Gray) |                               |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-------------------------------|
| Byte                      | Field Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Byte            | Field Description             |
| 0                         | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 0               | Frame (0xFE)                  |
| 1, 2                      | Length (0xNNNN)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 1, 2            | Length (0xNNNN)               |
| 3, 4                      | Message Type (0x00DB)                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 3, 4            | Message Type (0x00DB)         |
| 5                         | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 5               | Reserved (0x00)               |
| 6                         | Sequence Number                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 6               | Same Sequence Number          |
| 7                         | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 7               | Logical Node ID               |
| :                         | <b>AIB</b><br>Address Method<br>0x00 - Individual AEs<br>Number of AEs to follow<br>AE<br>0x0C Logical Span                                                                                                                                                                                                                                                                                                                                                                                        | 8, 9            | Status MSB,LSB                |
| :                         | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 10              | <b>AIB</b><br>Same as message |
| Response continued below. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                 |                               |
| :                         | Format<br>This field is a bit mask. The meaning of each bit is described below.<br>Unless otherwise stated, 0 = Disabled, 1 = Enabled.<br><br>Coding Method<br>Bit 0<br>0 Alternate Mark Inversion (AMI)<br>1 High Density Bipolar 3 (HDB3)<br><br>Error Checking<br>Bit 1 CRC-4 Enabled<br>Bit 2 FEBE Enabled<br><br>Signaling Method<br>Bit 3 Clear Channel<br>Bit 4 Transmit all zeroes<br>Bit 5<br>0 E1 Layer I management (Default)<br>1 Euro-ISDN E1 Layer I management<br>Bits 6-7 Reserved |                 |                               |

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | Line Length<br>0x00 G.703 ITU-T (75 Ohm) (Default)<br>0x04 G.703 ITU-T (120 Ohm)                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| : | Line Card Slot Number                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| : | Span Offset                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| : | SA Register Contents<br>This field is a bit mask. The meaning of each bit is described as follows.<br>Unless otherwise stated, 0 = Disabled, 1 = Enabled.<br><br>Bits 0–2 Reserved (must be 0)<br>Bit 3 Value to which all bits in SA Bit 4 Register are set.<br>Bit 4 Value to which all bits in SA Bit 5 Register are set.<br>Bit 5 Value to which all bits in SA Bit 6 Register are set.<br>Bit 6 Value to which all bits in SA Bit 7 Register are set.<br>Bit 7 Value to which all bits in SA Bit 8 Register are set. |
| : | TS 16 Register Contents<br>This field is a bit mask. The meaning of each bit is described below.<br>Unless otherwise stated, 0 = Disabled, 1 = Enabled.<br><br>Bits 0–3 Reserved (must be 0)<br>Bit 4 Value to which spare bit x 1 in frame 0 of timeslot 16 is set.<br>Bit 5 Reserved (must be 0)<br>Bit 6 Value to which spare bit x 3 in frame 0 of timeslot 16 is set.<br>Bit 7 Value to which spare bit x 4 in frame 0 of timeslot 16 is set.                                                                        |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

### NOTES

1. The Dialogic message bit-numbering format is 0–7, while the SA Register and TS 16 Register formats are 1–8. Therefore, bit 3 of the host message format is bit 4 of the SA Register and TS 16 Register formats, and so on.
2. This message is backward-compatible to previous software releases. If using a System Software 3.X Version 3.09 or earlier, the SA Register and TS 16 Register bits will be set to the default (1), and an *E1 Span Query* message will return a value of 0xF8 for the SA Register byte and 0xD0 for the TS 16 Register byte.

# EXNET Ring Configure 0x0074

---

|                       |                                                                                                                                                                                                                                                                                                                                                |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | RingConfig                                                                                                                                                                                                                                                                                                                                     |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                                                                                                                                                                                                                                              |
| <b>Description</b>    | <p><b>EXNET Ring Configure 0x0074</b></p> <p>Use this message to configure the EXNET® ring.<br/>The configuration includes:</p> <ul style="list-style-type: none"> <li>• Logical Ring ID</li> <li>• EXNET-ONE Card Diagnostics</li> <li>• Prepare For Addition</li> <li>• Expand Ring</li> <li>• Loop Back Port</li> <li>• Add Node</li> </ul> |
| <b>Sent by</b>        | Host                                                                                                                                                                                                                                                                                                                                           |

## SwitchKit Code Configuration

```
RingConfig (
 Node = integer,
 Slot = integer,
 Ring = integer,
 Entity = integer,
 Data = byte array);
```

## C Structure

```
typedef struct {
 UBYTE AddrInfo[30];
 UBYTE Entity;
 UBYTE Data[222];
} XL_RingConfig;
```

## C++ Class

```
class XLC_RingConfig : public XLC_OutboundMessage {
public:
 const UBYTE *getAddrInfo() const;
 UBYTE *getAddrInfo();
 void setAddrInfo(UBYTE *x);
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
 UBYTE getEntity() const;
 void setEntity(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

**Overview of message** The following table provides an overview of this message. The table following it, provides the detail for each byte.

| MESSAGE (White) |                            | RESPONSE (Gray) |                       |
|-----------------|----------------------------|-----------------|-----------------------|
| Byte            | Field Description          | Byte            | Field Description     |
| 0               | Frame (0xFE)               | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)            | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x0074)      | 3, 4            | Message Type (0x0074) |
| 5               | Reserved (0x00)            | 5               | Reserved (0x00)       |
| 6               | Sequence Number            | 6               | Same Sequence Number  |
| 7               | Logical Node ID            | 7               | Logical Node ID       |
| :               | AIB (starting with byte 0) | 8, 9            | Status MSB, LSB       |
| :               | Entity                     | 10              | Checksum              |
| :               | Data                       |                 |                       |
| :               | :                          |                 |                       |
| :               | Checksum                   |                 |                       |

**EXS API Hex Format -  
Detailed**

| MESSAGE (White) |                       | RESPONSE (Gray) |                       |
|-----------------|-----------------------|-----------------|-----------------------|
| Byte            | Field Description     | Byte            | Field Description     |
| 0               | Frame (0xFE)          | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)       | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x0074) | 3, 4            | Message Type (0x0074) |
| 5               | Reserved (0x00)       | 5               | Reserved (0x00)       |
| 6               | Sequence Number       | 6               | Same Sequence Number  |
| 7               | Logical Node ID       | 7               | Logical Node ID       |

EXNET Ring Configure 0x0074

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |      |                                                                                                                                                                                                                                                                                                                                                                                          |
|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 8, 9 | Status MSB, LSB<br>If the MSB of this field is 0x50 (NACK), the LSB indicates the NACK reason.<br><br>0x47 Invalid Ring Loop Timing<br>This value is returned when trying to set a node to be master-configurable, if the ring timing has already been derived.<br><br>0xC0 Action Denied<br>A value of Action Denied (0xC0) is returned if the user is trying to set an EXNET-ONE card. |
|   | Number of AEs to follow<br>AEs<br>0x01 Slot<br>to configure these entities:<br><br>0x01 Logical Ring ID<br>0x04 EXNET T-ONE Card<br>Diagnostics<br>0x05 Prepare for Addition<br>0x06 Expand Ring<br>0x07 Loopback Port<br>0x08 Add Note<br><br>OR<br>0x1A EXNET Ring<br>to configure these entities:<br><br>0x02 Transmit Mode<br>0x03 Number of Packets<br>0x04 EXNET-ONE Card Diagnostics<br>0x05 Prepare For Addition<br>0x06 Expand Ring<br>0x07 Loop Back Port<br>0x08 Add Node<br>0x09 Master-configurable<br>0x0A Ring Mode | 10   | Checksum                                                                                                                                                                                                                                                                                                                                                                                 |
| : | Entity<br>0x01 Logical Ring ID<br>0x02 Transmit Mode<br>0x03 Configure Number of Packets<br>0x04 EXNET-ONE Card Diagnostics<br>0x05 Prepare For Addition<br>0x06 Expand Ring<br>0x07 Loop Back Port<br>0x08 Add Node<br>0x09 Master-configurable<br>0x0A Ring Mode                                                                                                                                                                                                                                                                 |      |                                                                                                                                                                                                                                                                                                                                                                                          |
| : | Data                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |      |                                                                                                                                                                                                                                                                                                                                                                                          |

0x01 Assign Ring

Data[0] Logical Ring ID

De-assigning an EXNET Ring

Use the values below to de-assign a specific ring or all rings on a node:

0x02 Configure Transmit Mode

Data[0] Transmit Mode

0x00 Receive Only

0x01 Transmit and Receive

0x03 Configure Number of Packets

Data[0] Number of Packets

0x01 - No longer used.

0x03 - No longer used.

0x04 - Four packets (E1, J1, VDAC-ONE, IP Network Interface Series 2, and PCI H.100 EXNET Connect nodes).

0x04 EXNET-ONE Card Diagnostics

Data[0] EXNET-ONE Card Diagnostics

0x00 Reserved

0x01 Non-Forced\*, Internal Loop Back

0x02 Non-Forced\*, External\*\* Loop Back

0x03 Forced, Internal Loop Back

0x04 Forced, External\*\* Loop Back

\*The Non-Forced mode is used to prevent execution of EXNET-ONE Card Diagnostics procedure while EXNET® Ring is up and running. Performing EXNET-ONE Card Diagnostics will take the EXNET® Ring out of service.

\*\*The External Loop Back mode can be used when both I/O ports are externally looped back with fiber optic cabling.

0x05 Prepare For Addition

There is no data associated with this entity.

0x06 Expand Ring

There is no data associated with this entity.

0x07 Loop Back Port

Node ID of the neighbor CSP node. This message must be sent to the node that will loopback its "A" port. The neighbor node will automatically loopback "B" port.

Data[0] 0x00 Logical Node ID of node whose "B" port will be looped back.

0x08 Add Node

There is no data associated with this entity.

EXNET Ring Configure 0x0074

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   | <p>0x09 Master-configurable</p> <p>The value of this field indicates whether or not a node is allowed to become the EXNET® Ring Master. Valid entries for these fields, based on the value of the <i>Entity</i> field value 0x09, are as follows:</p> <p style="padding-left: 20px;">Data[0] Allow Master Arbitration Flag.</p> <p style="padding-left: 40px;">0x00 Do not allow this node to participate in EXNET® Ring Master Arbitration.</p> <p style="padding-left: 40px;">0x01 Allow this node to participate in EXNET® Ring Master Arbitration. (Default for CSP 2090 and CSP 2040 nodes)</p> <p>0x0A Ring Mode</p> <p>The Ring Mode entity configures the Enhanced Fault Tolerance Ring Mode.</p> <p style="padding-left: 20px;">Data[0] Ring Mode</p> <p style="padding-left: 40px;">0x00 Reserved</p> <p style="padding-left: 40px;">0x01 Enhanced Fault Tolerance Ring Mode</p> <p style="padding-left: 60px;">This mode adds enhanced Single Ring Redundancy with seamless switchover support, run-time software link validation, enhanced live node insertion, and overall improved performance and stability. This mode is recommended for homogenous second generation EXNET® rings.</p> |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

| Field                    | Field Value to Deassign a Specific Ring | Field Value to Deassign All Rings |
|--------------------------|-----------------------------------------|-----------------------------------|
| AIB                      | Slot #                                  | 0xFF                              |
| Entity                   | 0x01 (Assign Ring)                      |                                   |
| Data[0]– Logical Ring ID | 0xFF                                    |                                   |

## EXS Node Configure 0x007F

---

|                       |                                                                                       |
|-----------------------|---------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | NodeConfig                                                                            |
| <b>Type</b>           | EXS API and SwitchKit API message                                                     |
| <b>Description</b>    | <b>EXS Node Configure 0x007F</b><br>Use this message to configure entities in a node. |
| <b>Sent by</b>        | Host                                                                                  |
| <b>SwitchKit Code</b> | <b>Configuration</b>                                                                  |

```
NodeConfig (
 Node = integer,
 Entity = integer,
 DataLen = integer,
 Data = byte array);
```

### C Structure

```
typedef struct {
 unsigned short Entity;
 UBYTE DataLen;
 UBYTE Data[250];
} XL_NodeConfig;
```

### C Structure Response

```
typedef struct {
 unsigned short Status;
 unsigned short Entity;
 UBYTE DataLength;
 UBYTE Data[248];
} XL_NodeConfigAck;
```

### C++ Class

```
class XLC_NodeConfig : public XLC_OutboundMessage {
public:
 unsigned short getEntity() const;
 void setEntity(unsigned short x);
 UBYTE getDataLen() const;
 void setDataLen(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

### C++ Class Response

```
class XLC_NodeConfigAck : public XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
```

```

unsigned short getEntity() const;
void setEntity(unsigned short x);
UBYTE getDataLength() const;
void setDataLength(UBYTE x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};

```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                                                                                           | RESPONSE (Gray) |                                                                                                                                                                                                                                                                                                                                                              |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                                                                                                         | Byte            | Field Description                                                                                                                                                                                                                                                                                                                                            |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                              | 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                 |
| 1, 2            | Length (0xNNNN)                                                                                                                                                                                                                                                                                                                                                           | 1, 2            | Length (0xNNNN)                                                                                                                                                                                                                                                                                                                                              |
| 3, 4            | Message Type (0x007F)                                                                                                                                                                                                                                                                                                                                                     | 3, 4            | Message Type (0x007F)                                                                                                                                                                                                                                                                                                                                        |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                           | 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                              |
| 6               | Sequence Number                                                                                                                                                                                                                                                                                                                                                           | 6               | Same Sequence Number                                                                                                                                                                                                                                                                                                                                         |
| 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                                                           | 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                                              |
| 8, 9            | Entity<br>0x0001 Number of Timeslots<br>0x0002 Reserved<br>0x0003 Number of EXNET Connect Channels<br>0x0004 Clock Master<br>0x0005 Master on Multiple Ring Configure<br>0x0006 CTNETREF/PCI H.100                                                                                                                                                                        | 8, 9            | Status MSB, LSB                                                                                                                                                                                                                                                                                                                                              |
| 10              | Data Length                                                                                                                                                                                                                                                                                                                                                               | 10, 11          | Entity                                                                                                                                                                                                                                                                                                                                                       |
| 11              | Data<br>The meaning of this field depends on the value of the Entity field.<br><br>0x0001 Number of Timeslots<br>Data Length: 0x02<br><br>The value of this word-length field represents the number of timeslots assigned for CSP Conferencing. The valid entry for this field is as follows:<br><br>Data[0,1] 0xFFFF<br>Assign all unused timeslots<br><br>0x0002 Unused | 12              | Data Length                                                                                                                                                                                                                                                                                                                                                  |
|                 |                                                                                                                                                                                                                                                                                                                                                                           | 13              | Data<br>The meaning of these fields depends on the value of the Entity field in the message.<br><br>0x0001 Number of Timeslots<br>Valid entries for these fields based on the Entity field value 0x01 are as follows:<br><br>Data Length: 0x02<br>Data[0,1] Number of Timeslots<br>Allocated for CSP Conferencing Conferencing (MSB, LSB)<br><br>0x02 Unused |
|                 |                                                                                                                                                                                                                                                                                                                                                                           | :               | Checksum                                                                                                                                                                                                                                                                                                                                                     |

EXS Node Configure 0x007F

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   | <p>0x03 Number of EXNET Connect Channels<br/>           Data Length: 0x03<br/>           Data[0, 1] Span (MSB, LSB) All spans 0xFFFF<br/>           Data[2] Number of Channels<br/>               0x18 - 24 (T1)<br/>               0x20 - 32 (E1)</p> <p>Note: You must de-assign spans before modifying the number of channels per span.</p> <p>0x04 Clock Master<br/>           Data Length: 0x03</p> <p>Data[0] Slot Number<br/>           Data[1] Bus Master<br/>               0x00 Not SCbus clock master/H.100 Slave to clock A<br/>               0x01 Become SCbus clock master/H.100 primary clock masterdriving clock A<br/>               0x02 H.100 secondary clock master driving clock A<br/>               0x03 H.100 slave to clock B<br/>               0x04 H.100 primary clock master driving clock B<br/>               0x05 H.100 secondary clock master driving clock B</p> <p>Data[2] Speed<br/>               0x00 4 MHz<br/>               0x01 2 MHz (Reserved)<br/>               0x02 8 MHz</p> <p>Example<br/>           Data[2] corresponds to the local bus speed. Value 0x00 corresponds to SCSA compatibility mode speed for PCI/H.100 card.</p> <p>0x05 Master on Multiple Ring Configure<br/>           Indicates whether or not a node is allowed to become the EXNET® Master of multiple rings.<br/>           Data Length: 0x01<br/>           Data[0] Master on Multiple Ring Flag<br/>               0x01 Allow this node to be Master on more than one ring simultaneously.<br/>               0x00 Do not allow this node to be Master on more than one ring simultaneously. (Default)</p> <p>0x06 CTNETREF<br/>           Data[0] Slot Number</p> |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

## EXS Node Configuration Query 0x007E

---

|                       |                                                                                                                   |
|-----------------------|-------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | NodeConfigQuery                                                                                                   |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                 |
| <b>Description</b>    | <b>EXS Node Configuration Query 0x007E</b><br>Use this message to query various configuration entities in a node. |
| <b>Sent by</b>        | Host                                                                                                              |
| <b>SwitchKit Code</b> | <b>C Structure</b>                                                                                                |

```
typedef struct {
 unsigned short Entity;
 UBYTE DataLen;
 UBYTE Data[250];
} XL_NodeConfigQuery;
```

### C Structure Response

```
typedef struct {
 unsigned short Status;
 unsigned short Entity;
 UBYTE DataLength;
 UBYTE Data[248];
} XL_NodeConfigQueryAck;
```

### C++ Class

```
class XLC_NodeConfigQuery : public XLC_OutboundMessage {
public:
 unsigned short getEntity() const;
 void setEntity(unsigned short x);
 UBYTE getDataLen() const;
 void setDataLen(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

### C++ Class Response

```
class XLC_NodeConfigQueryAck : public
 XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 unsigned short getEntity() const;
 void setEntity(unsigned short x);
 UBYTE getDataLength() const;
 void setDataLength(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
```

};

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                             | RESPONSE (Gray) |                                                                                                                                                   |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| Byte            | Field Description                                                                                                                                           | Byte            | Field Description                                                                                                                                 |
| 0               | Frame (0xFE)                                                                                                                                                | 0               | Frame (0xFE)                                                                                                                                      |
| 1, 2            | Length (0xNNNN)                                                                                                                                             | 1, 2            | Length (0xNNNN)                                                                                                                                   |
| 3, 4            | Message Type (0x007E)                                                                                                                                       | 3, 4            | Message Type (0x007E)                                                                                                                             |
| 5               | Reserved (0x00)                                                                                                                                             | 5               | Reserved (0x00)                                                                                                                                   |
| 6               | Sequence Number                                                                                                                                             | 6               | Same Sequence Number                                                                                                                              |
| 7               | Logical Node ID                                                                                                                                             | 7               | Logical Node ID                                                                                                                                   |
| 8, 9            | Entity MSB, LSB<br>0x0001 Number of Timeslots<br>0x0002 Reserved<br>0x0004 Clock Master<br>0x0005 Master on Multiple Ring<br>Configure<br>0x0006 - CTNETREF | 8, 9            | Status (MSB, LSB)                                                                                                                                 |
| 10              | Data Length                                                                                                                                                 | 10, 11          | Entity (MSB, LSB)                                                                                                                                 |
|                 |                                                                                                                                                             | 12              | Data Length                                                                                                                                       |
| 11              | Data<br><br>0x01 Number of Timeslots<br>Data[0,1] Number of<br>Timeslots assigned for CSP<br>Conferencing (Enter 0xFFFF<br>to assign all unused timeslots)  | 13              | Data<br>0x01 Number of Timeslots<br>Data[0,1] Number of timeslots<br>assigned<br><br>Data[2,3] Number of timeslots that<br>are actually allocated |
|                 |                                                                                                                                                             | 14              | Checksum                                                                                                                                          |

0x04 Clock Master

Data[0] Slot Number

Data[1] Configured Bus Master

- 0x00 PCI H.100 EXNET Connect is configured to be slave to clock A of H.100 local bus.
- 0x01 H.100 primary clock master driving clock A PCI H.100 EXNET Connect is configured to be H.100 local bus master driving clock A.
- 0x02 H.100 secondary clock master driving clock. This is clock A.
- 0x03 H.100 slave to clock B
- 0x04 H.100 primary clock master driving clock B
- 0x05 H.100 secondary clock master driving clock B

Data[2] Configured Speed

- 0x00 4 MHz [SCSA compatibility mode] EXNET Connects are configured at this bus speed. Lower 16 data lines of H.100 bus are configured at this bus speed.
- 0x02 8 MHz [Full H.100 mode] All 32 data lines of H.100 bus are configured at this bus speed.

Data[3] Actual Bus Master

- 0x00 H.100 Slave to clock A. EXNET Connect actual state is not as local bus master. PCI H.100 EXNET Connect actual state is slave to clock A of H.100 local bus.
- 0x01 H.100 primary clock master driving clock. A PCI H.100 EXNET Connect actual state is H.100 local bus master driving clock A.
- 0x02 H.100 secondary clock master driving clock A
- 0x03 H.100 slave to clock B
- 0x04 H.100 primary clock master driving clock B
- 0x05 H.100 secondary clock master driving clock B
- 0x06 H.100 bus not available

Data[4] Actual Speed

- 0x00 4 MHz [SCSA compatibility mode] EXNET Connects are at this bus speed. Lower 16 data lines of H.100 bus are at this bus speed.
- 0x02 8 MHz [Full H.100 mode] All 32 data lines of H.100 bus are at this bus speed.

EXS Node Configuration Query 0x007E

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   | <p>0x05 Master On Multiple Ring Configure<br/>         The value of this field indicates whether or not a node is allowed to become the EXNET® Master of multiple rings.</p> <p>Data[0] Master on Multiple Ring Flag<br/>         0x00 Do not allow this node to be Master on more than one ring simultaneously (Default)<br/>         0x01 Allow this node to be Master on more than one ring simultaneously.</p> <p>0x06 CTNETREF<br/>         Data[0] Slot Number (Indicates slot number of the board that is queried.)</p> <p>Data[1] CTNETREF Drive/Receive<br/>         0x00 Receive. Indicates the board is configured to receive the CTNETREF signal from other boards on the H.100 bus.<br/>         0x01 Drive</p> <p>Data[2] CTNETREF Speed<br/>         0x00 - 2.048 MHz<br/>         0x01 - 1.54 MHz<br/>         0x02 - 8 kHz</p> |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## Fault Log Query 0x0086

---

**SwitchKit Name** FaultLogQuery

**Type** EXS API and SwitchKit API message

**Description** **Fault Log Query 0x0086**

This message should be sent by the host if it receives a *Card Status Report* message or a response to a *Card Status Query* message with the “Faults Logged” bit set. Faults are queried for the slot specified. The host receives separate responses for each fault logged. There are two types of logs: a fault log containing a log that can be cleared after it is reported by setting the Action field to 0x02. The response data should be reported to Dialogic Technical Support for evaluation.

The host can also send this message to the active or standby CSP 2000 Matrix Card.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 UBYTE AddrInfo[30];
 UBYTE Action;
} XL_FaultLogQuery;
```

**C Structure Response**

```
typedef struct {
 unsigned short Status;
 UBYTE AddrInfo[30];
 UBYTE NumFaults;
 UBYTE FaultNumber;
 char FaultBuffer[80];
} XL_FaultLogQueryAck;
```

**C++ Class**

```
class XL_C_FaultLogQuery : public XLC_OutboundMessage {
public:
 const UBYTE *getAddrInfo() const;
 UBYTE *getAddrInfo();
 void setAddrInfo(UBYTE *x);
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
 UBYTE getModule() const;
 void setModule(UBYTE x);
 UBYTE getAction() const;
 void setAction(UBYTE x);
};
```

### C++ Class Response

```
class XLC_FaultLogQueryAck : public
 XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 const UBYTE *getAddrInfo() const;
 UBYTE *getAddrInfo();
 void setAddrInfo(UBYTE *x);
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
 UBYTE getModule() const;
 void setModule(UBYTE x);
 UBYTE getNumFaults() const;
 void setNumFaults(UBYTE x);
 UBYTE getFaultNumber() const;
 void setFaultNumber(UBYTE x);
 const char *getFaultBuffer() const;
 void setFaultBuffer(const char *x);
};
```

### EXS API Hex Format

| MESSAGE (White) |                                                                                                                         | RESPONSE (Gray) |                                                                                                                                                                                                                                                                                                   |
|-----------------|-------------------------------------------------------------------------------------------------------------------------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Byte            | Field Description                                                                                                       | Byte            | Field Description                                                                                                                                                                                                                                                                                 |
| 0               | Frame (0xFE)                                                                                                            | 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                      |
| 1, 2            | Length (0xNNNN)                                                                                                         | 1, 2            | Length (0xNNNN)                                                                                                                                                                                                                                                                                   |
| 3, 4            | Message Type (0x0086)                                                                                                   | 3, 4            | Message Type (0x0086)                                                                                                                                                                                                                                                                             |
| 5               | Reserved (0x00)                                                                                                         | 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                   |
| 6               | Sequence Number                                                                                                         | 6               | Same Sequence Number                                                                                                                                                                                                                                                                              |
| 7               | Logical Node ID                                                                                                         | 7               | Logical Node ID                                                                                                                                                                                                                                                                                   |
| :               | AIB<br>Address Method<br>0x00 - Individual AEs<br>Number of AEs to follow<br>AEs<br>0x01 Slot<br>or<br>0x42 VoIP Module | 8, 9            | Status MSB, LSB<br>If the value of this field <b>is</b> a positive acknowledgment (0x0010), then the <i>Fault Data</i> bytes apply.<br><br>If the value of this field <b>is not</b> a positive acknowledgment (0x0010), then the <i>Fault Data</i> bytes are not reported and the length is 0x07. |
| :               | Action<br>0x01 Return Fault Log<br>0x02 Return Fault Log,<br>then clear Fault Log<br>0x03 Return Stack Trace            | 10              | AIB (starting with Byte 0)                                                                                                                                                                                                                                                                        |
| :               | Checksum                                                                                                                |                 |                                                                                                                                                                                                                                                                                                   |

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   | Response continued below.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| : | <p><b>Fault Data</b><br/>The response data varies depending on the type of fault log queried.</p> <p>0x01 or 0x02 <b>Fault Log</b><br/> <b>Data[0]</b> Number of Faults<br/> Indicates the total number of faults in the log.</p> <p><b>Data[1]</b> Fault Number<br/> The first response is the number of the last fault logged.<br/> The last response is fault Number 1.</p> <p><b>Data[2+] Data</b><br/> A null-terminated text string with a maximum of 128 bytes.<br/> If the text string is less, it will be terminated with 0x0A.</p> <p>0x03 <b>Fault Log and Trace</b><br/> <b>Data[0]</b> Reserved (0x00)</p> <p><b>Data[1]</b> Reserved (0x00)</p> <p><b>Data[2+]</b> A NULL terminated ASCII string, followed by the stack pointer when the fault occurred, followed by a trace of the stack.</p> |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

**NOTES:**

1. Faults are logged as NULL terminated readable text strings. The host will receive a message for each fault in the log.
2. When in any state other than active, the only valid slot is the matrix that the message is sent to. All requests for other slots will receive a response status of negative acknowledgment.

# Filter/Timer Configure 0x0012

---

**SwitchKit Name** FilterTimerConfig

**Type** EXS API and SwitchKit API message

**Description** **Filter/Timer Configure 0x0012**

This message is used to adjust trunk and line signaling filters and timers. Typically, it is only required when the trunk or line that the CSP is connected to is not operating to line interface specifications.

All timers and filters are set to default values based on industry specifications when the channel trunk type is configured. To query the timers and filters use the *Channel Parameter Query* message.

**NOTE:** Use this message to configure the PPL timers for all STILC line cards. You cannot use the *PPL Timer Configure* message with the STILC line cards.

**Sent by** Host

**Overview of message** The following table provides an overview of this message. The table following it, provides the detail for each byte.

**SwitchKit Code** **Configuration**

```
FilterTimerConfig (
 Node = integer,
 Range = StartSpan:StartChan - EndSpan:EndChan,
 Type = integer,
 FilterID = integer,
 Value = integer);
```

**C Structure**

```
typedef struct {
 unsigned short StartSpan;
 UBYTE StartChannel;
 unsigned short EndSpan;
 UBYTE EndChannel;
 UBYTE Type;
 UBYTE FilterID;
 unsigned short Value;
} XL_FilterTimerConfig;
```

**C++ Class**

```
class XLC_FilterTimerConfig : public XLC_ChanRangeMessage
{
public:
 unsigned short getStartSpan() const;
 void setStartSpan(unsigned short x);
 UBYTE getStartChannel() const;
```

```

void setStartChannel(UBYTE x);
unsigned short getEndSpan() const;
void setEndSpan(unsigned short x);
UBYTE getEndChannel() const;
void setEndChannel(UBYTE x);
UBYTE getType() const;
void setType(UBYTE x);
UBYTE getFilterID() const;
void setFilterID(UBYTE x);
unsigned short getValue() const;
void setValue(unsigned short x);
};

```

**Overview of Message**

| MESSAGE (White) |                            | RESPONSE (Gray) |                       |
|-----------------|----------------------------|-----------------|-----------------------|
| Byte            | Field Description          | Byte            | Field Description     |
| 0               | Frame (0xFE)               | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)            | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x0012)      | 3, 4            | Message Type (0x0012) |
| 5               | Reserved (0x00)            | 5               | Reserved (0x00)       |
| 6               | Sequence Number            | 6               | Same Sequence Number  |
| 7               | Logical Node ID            | 7               | Logical Node ID       |
| :               | AIB (starting with byte 0) | 8, 9            | Status (MSB, LSB)     |
| :               | Filter/Timer Type          | 10              | Checksum              |
| :               | Filter/Timer ID            |                 |                       |
| :               | Value (MSB, LSB)           |                 |                       |
| :               | Checksum                   |                 |                       |

**EXS API Hex Format**

| MESSAGE (White) |                                                         | RESPONSE (Gray) |                       |
|-----------------|---------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                       | Byte            | Field Description     |
| 0               | Frame (0xFE)                                            | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)                                         | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x0012)                                   | 3, 4            | Message Type (0x0012) |
| 5               | Reserved (0x00)                                         | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                         | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                         | 7               | Logical Node ID       |
| :               | <u>AIB</u><br>Address Method<br>0x01 - Range of AEs     | 8, 9            | Status (MSB, LSB)     |
|                 | Number of AEs to follow<br>0x02                         |                 |                       |
|                 | AEs<br>0x0D Channel (Starting)<br>0x0D Channel (Ending) |                 |                       |

Filter/Timer Configure 0x0012

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |    |          |
|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----------|
| : | Filter/Timer Type<br>0x01 Signal Scanning Filter<br>0x02 Signal Scanning Timer<br>0x03 Transmit Signaling Timer                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 10 | Checksum |
| : | Filter/Timer ID<br>This field indicates the specific timer or filter. The meaning of the ID number depends on the value of the <i>Filter/Timer Type</i> field. Valid entries for this field are listed below by <i>Filter/Timer Type</i> field value:<br><br>0x01 Signal Scanning Filter<br>0x01 Preseize<br>0x02 Inseize<br>0x03 Modified Incoming Release<br>0x04 Modified Outgoing Release<br>0x05 Normal Incoming Release<br>0x06 Normal Incoming Release with Flash<br>0x07 Normal Outgoing Release<br>0x08 Normal Outgoing Release with Flash<br>0x09 Post Inseize Acknowledgment<br>0x0A Outseize Acknowledgment<br>0x0B Outseize Answer<br>0x0C Outseize Dial Signal End<br>0x0D First Release<br>0x0E ANI Request Off-hook |    |          |

## Filter/Timer Configure 0x0012

|      |                                     |
|------|-------------------------------------|
| 0x02 | Signal Scanning Timer               |
| 0x01 | Inseize Complete                    |
| 0x02 | Post Inseize Complete               |
| 0x03 | Start Normal Outgoing Release       |
| 0x04 | Start Normal Incoming Release       |
| 0x05 | Outseizure Acknowledgment           |
| 0x06 | Outseizure Answer                   |
| 0x07 | Guard Timeout                       |
| 0x08 | Release                             |
| 0x09 | Glare Detection                     |
| 0x0A | Minimum Flash                       |
| 0x0B | Maximum Flash                       |
| 0x0C | Start Dial Minimum Wink             |
| 0x0D | Start Dial Maximum Wink             |
| 0x0E | Minimum Delay Dial Signal           |
| 0x0F | Maximum Delay Dial Signal           |
| 0x10 | Post Start Dial Signal Output Delay |
| 0x11 | Wink 2 Minimum Receive Duration     |
| 0x12 | Wink 2 Maximum Receive Duration     |
| 0x13 | Wink 2 Minimum Post Output Delay    |
| 0x14 | Wink 2 Maximum Post Output Delay    |
| 0x15 | Wink 2 Maximum Receive Detection    |
| 0x16 | Wink 3 Minimum Receive Duration     |
| 0x17 | Wink 3 Maximum Receive Duration     |
| 0x18 | Wink 3 Minimum Post Output Delay    |
| 0x19 | Wink 3 Maximum Post Output Delay    |
| 0x1A | Wink 3 Maximum Receive Detection    |
| 0x1B | Wink 4 Minimum Receive Duration     |
| 0x1C | Wink 4 Maximum Receive Duration     |
| 0x1D | Wink 4 Minimum Post Output Delay    |
| 0x1E | Wink 4 Maximum Post Output Delay    |
| 0x1F | Wink 4 Maximum Receive Detection    |
| 0x20 | Wink 5 Minimum Receive Duration     |
| 0x21 | Wink 5 Maximum Receive Duration     |
| 0x22 | Wink 5 Minimum Post Output Delay    |
| 0x23 | Wink 5 Maximum Post Output Delay    |
| 0x24 | Wink 5 Maximum Receive Detection    |
| 0x25 | Wink 6 Minimum Receive Duration     |
| 0x26 | Wink 6 Maximum Receive Duration     |
| 0x27 | Wink 6 Minimum Post Output Delay    |
| 0x28 | Wink 6 Maximum Post Output Delay    |
| 0x29 | Wink 6 Maximum Receive Detection    |
| 0x2A | Wink 7 Minimum Receive Duration     |
| 0x2B | Wink 7 Maximum Receive Duration     |
| 0x2C | Wink 7 Minimum Post Output Delay    |
| 0x2D | Wink 7 Maximum Post Output Delay    |
| 0x2E | Wink 7 Maximum Receive Detection    |
| 0x2F | Wink 8 Minimum Receive Duration     |
| 0x30 | Wink 8 Maximum Receive Duration     |
| 0x31 | Wink 8 Minimum Post Output Delay    |
| 0x32 | Wink 8 Maximum Post Output Delay    |

Filter/Timer Configure 0x0012

|   |                                                      |
|---|------------------------------------------------------|
|   | 0x33 Wink 8 Maximum Receive Detection                |
|   | 0x34 Post ANI Off-hook Outpulse Delay                |
|   | 0x35 Maximum Receive ANI Off-hook Request Detection  |
|   | 0x36 Maximum Receive Dialtone Detection              |
|   | 0x37 MFR1 Minimum Receive KP Duration                |
|   | 0x38 MFR1 Minimum Receive Digit Duration             |
|   | 0x39 MFR1 Maximum Receive KP Duration                |
|   | 0x3A MFR1 Minimum Receive Interdigit Duration        |
|   | 0x3B MFR1 Maximum Receive Interdigit Duration        |
|   | 0x3C DTMF Maximum Receive 1st Digit Detection        |
|   | 0x3D DTMF Minimum Receive Digit Duration             |
|   | 0x3E DTMF Minimum Receive Interdigit Duration        |
|   | 0x3F DTMF Maximum Receive Interdigit Duration        |
|   | 0x03 Transmit Signal Timer                           |
|   | 0x01 Wink Duration                                   |
|   | 0x02 Flash Duration                                  |
|   | 0x03 Primary Outseizure Signal                       |
|   | 0x04 Outseize Signaling Complete                     |
|   | 0x05 Delay Dial Signal Start                         |
|   | 0x06 Fixed Pause                                     |
|   | 0x07 Timed Answer                                    |
|   | 0x08 Ringing On Duration                             |
|   | 0x09 Ringing Off Duration                            |
|   | 0x0A Wink 1 Minimum Transmit Delay                   |
|   | 0x0B Wink 2 Minimum Transmit Delay                   |
|   | 0x0C Wink 2 Maximum Transmit Delay                   |
|   | 0x0D Wink 2 Transmit Duration                        |
|   | 0x0E Wink 3 Minimum Transmit Delay                   |
|   | 0x0F Wink 3 Maximum Transmit Delay                   |
|   | 0x10 Wink 3 Transmit Duration                        |
|   | 0x11 Wink 4 Minimum Transmit Delay                   |
|   | 0x12 Wink 4 Maximum Transmit Delay                   |
|   | 0x13 Wink 4 Transmit Duration                        |
|   | 0x14 Wink 5 Minimum Transmit Delay                   |
|   | 0x15 Wink 5 Maximum Transmit Delay                   |
|   | 0x16 Wink 5 Transmit Duration                        |
|   | 0x17 Wink 6 Minimum Transmit Delay                   |
|   | 0x18 Wink 6 Maximum Transmit Delay                   |
|   | 0x19 Wink 6 Transmit Duration                        |
|   | 0x1A Wink 7 Minimum Transmit Delay                   |
|   | 0x1B Wink 7 Maximum Transmit Delay                   |
|   | 0x1C Wink 7 Transmit Duration                        |
|   | 0x1D Wink 8 Minimum Transmit Delay                   |
|   | 0x1E Wink 8 Maximum Transmit Delay                   |
|   | 0x1F Wink 8 Transmit Duration                        |
|   | 0x20 MFR1 Transmit KP Duration                       |
|   | 0x21 MFR1 Transmit Digit Duration                    |
|   | 0x22 MFR1 Transmit Inter-digit Duration              |
|   | 0x23 DTMF Transmit Digit Duration                    |
|   | 0x24 DTMF Transmit Inter-digit Duration              |
| : | Value (MSB, LSB) Filter/Timer length in 10 ms units. |
| : | Checksum                                             |

## Flash Timing Configure 0x0016

---

|                       |                                                                                                                                                  |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | FlashTimingConfig                                                                                                                                |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                                                |
| <b>Description</b>    | <b>Flash Timing Configure 0x0016</b><br>Use this message to enable or disable flash detection and propagation on a channel or range of channels. |
| <b>Sent by</b>        | Host                                                                                                                                             |
| <b>SwitchKit Code</b> | <b>Configuration</b>                                                                                                                             |

```
FlashTimingConfig (
 Node = integer,
 Range = StartSpan:StartChan - EndSpan:EndChan,
 FlashMode = integer);
```

### C Structure

```
typedef struct {
 unsigned short StartSpan;
 UBYTE StartChannel;
 unsigned short EndSpan;
 UBYTE EndChannel;
 UBYTE FlashMode;
} XL_FlashTimingConfig;
```

### C++ Class

```
class XLC_FlashTimingConfig : public XLC_ChanRangeMessage
{
public:
 unsigned short getStartSpan() const;
 void setStartSpan(unsigned short x);
 UBYTE getStartChannel() const;
 void setStartChannel(UBYTE x);
 unsigned short getEndSpan() const;
 void setEndSpan(unsigned short x);
 UBYTE getEndChannel() const;
 void setEndChannel(UBYTE x);
 UBYTE getFlashMode() const;
 void setFlashMode(UBYTE x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                 | RESPONSE (Gray) |                       |
|-----------------|-----------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                               | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                    | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)                                                                                                 | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x0016)                                                                                           | 3, 4            | Message Type (0x0016) |
| 5               | Reserved (0x00)                                                                                                 | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                 | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                 | 7               | Logical Node ID       |
| :               | <u>AIB</u>                                                                                                      | 8, 9            | Status (MSB, LSB)     |
|                 | Address Method                                                                                                  | 10              | Checksum              |
|                 | 0x01 Range of AEs                                                                                               |                 |                       |
|                 | Number of AEs to follow                                                                                         |                 |                       |
| :               | AEs                                                                                                             |                 |                       |
|                 | 0x0D Channel (Starting)                                                                                         |                 |                       |
|                 | 0x0D Channel (Ending)                                                                                           |                 |                       |
| :               | Flash Mode                                                                                                      |                 |                       |
|                 | 0x00 Flash Timing OFF                                                                                           |                 |                       |
|                 | 0x01 Flash Timing ON, propagate Flash to distant end                                                            |                 |                       |
|                 | 0x02 Flash Timing ON, send <i>Call Processing Event</i> message to the host.                                    |                 |                       |
|                 | 0x03 Flash Timing ON, send <i>Call Processing Event</i> message to the host and propagate Flash to distant end. |                 |                       |
| :               | Checksum                                                                                                        |                 |                       |

# ForceGroupState

---

**Type** EXS SwitchKit API message

**Purpose** This message causes the LLC to take action on all of the channels within the group specified by the ChannelGroup field. This will only affect channels not currently owned or watched by another application.

**Sent by** SwitchKit Application

**Arguments** The following table shows the arguments you can change:

| Argument     | Description                                                                                                                                         |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| ChannelGroup | The name of the group this action should be performed on.                                                                                           |
| action       | The state the LLC is requesting this channel be put into. Values for ForceGroupState actions:<br><br>SK_OUT_OF_SERVICE =0x0F<br>SK_IN_SERVICE =0xF0 |
| int status   | This is a response argument. The values are:<br><br>SK_INVALID_GROUP - No valid channel group was found.<br>OK - Success                            |

## Response Arguments

| Argument     | Description                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------------------|
| ChannelGroup | The name of the group this action should be performed on.                                                     |
| int status   | The values for this argument are:<br><br>SK_INVALID_GROUP - No valid channel group was found.<br>OK - Success |

**C Structure**      typedef struct {  
                      char ChannelGroup[50];  
                      unsigned short Action;  
                      UBYTE reserved69[4];  
                      } *SK\_ForceGroupState*;

**C Structure Response**      typedef struct {  
                      char ChannelGroup[50];  
                      int Status;  
                      UBYTE reserved71[4];  
                      } *SK\_ForceGroupStateAck*;

**C++ Class**      class *SKC\_ForceGroupState* : public SKC\_ToolkitMessage {  
public:  
    const char \*getChannelGroup() const;  
    void setChannelGroup(const char \*x);  
    unsigned short getAction() const;  
    void setAction(unsigned short x);  
};

**C++ Class Response**      class *SKC\_ForceGroupStateAck* : public SKC\_ToolkitAck {  
public:  
    const char \*getChannelGroup() const;  
    void setChannelGroup(const char \*x);  
    int getStatus() const;  
    void setStatus(int x);  
};

## Generate Call Processing Event 0x00BA

---

|                       |                                                                                                                                                                                                                                    |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | GenerateCallProcessingEvent                                                                                                                                                                                                        |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                                                                                                                                  |
| <b>Description</b>    | <b>Generate Call Processing Event 0x00BA</b><br>Use this message to instruct the CSP to generate a call processing event on the specified channel. To generate external host PPL events, use the <i>PPL Event Request</i> message. |
| <b>Sent by</b>        | Host                                                                                                                                                                                                                               |

**Switchkit Code**    **C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
 UBYTE Event;
} XL_GenerateCallProcessingEvent;
```

**C++ Class**

```
class XLC_GenerateCallProcessingEvent : public
 XLC_OneChannelOutbound {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 UBYTE getEvent() const;
 void setEvent(UBYTE x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                                                                                     | RESPONSE (Gray) |                       |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                   | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                        | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)                                                                                                                                                                                                                                                                     | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x00BA)                                                                                                                                                                                                                                                               | 3, 4            | Message Type (0x00BA) |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                     | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                                                                                                                                                     | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                                                                                                                                                     | 7               | Logical Node ID       |
| :               | <b>AIB</b>                                                                                                                                                                                                                                                                          | 8, 9            | Status MSB, LSB       |
|                 | Address Method<br>0x00 - Individual AEs                                                                                                                                                                                                                                             | 10              | Checksum              |
|                 | Number of AEs to follow                                                                                                                                                                                                                                                             |                 |                       |
|                 | AE<br>0x0D Channel                                                                                                                                                                                                                                                                  |                 |                       |
| :               | <b>Event</b><br>0x01 Answer Call (Generate appropriate signaling to return answer to the incoming call)<br>This event might not be necessary if PPLevREQ(ANM) is sent.<br>0x02 Flash (Generate flash signaling to the incoming call)<br>0x11 Transmit Offhook<br>0x12 Transmit Idle |                 |                       |
| :               | Checksum                                                                                                                                                                                                                                                                            |                 |                       |

# Generic Card Configure 0x0122

---

**SwitchKit Name** GenericCardConfigure

**Type** EXS API and SwitchKit API message

**Description** **Generic Card Configure 0x0122**

Use this message for a variety of card configuration options.

## DSP Series 2 Card

For the DSP Series 2 card, use this message to configure the following:

- Card-level Defaults
- Alarms
- NFS location and Vocabulary Index File
- Sequence Number Size (8-bit or 16-bit)
- Advanced Conferencing Features
- T.30 Fax Parameters
- Positive Voice Detection/Answering Machine Detection
- Echo Cancellation Parameters
- Dial Pulse Detection Parameters

**Sent by** Host Application

**Related API Message** *Generic Card Query*

**SwitchKit Code** **C Structure**

```
typedef struct {
 UBYTE Slot;
 UBYTE reserved18[29];
 UBYTE DataType;
 UBYTE TLVCount;
 UBYTE Data[221];
} XL_GenericCardConfigure;
```

## C Structure Response

```
typedef struct {
 unsigned short Status;
 UBYTE reserved6[13];
 UBYTE TLVCount;
 UBYTE Data[250];
} XL_GenericCardConfigure;
```

### **C++ Class**

```
class XLC_GenericCardConfigure : public
 XLC_OutboundMessage {
public:
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
 UBYTE getDataType() const;
 void setDataType(UBYTE x);
 UBYTE getTLVCount() const;
 void setTLVCount(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

### **C++ Class Response**

```
class XLC_GenericCardConfigureAck : public
 XLC_OutboundMessage {
public:
 unsigned short getStatus() const
 void setStatus(unsigned short x)
 UBYTE getTLVCount() const;
 void setTLVCount(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x)
 ;
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                       | RESPONSE (Gray) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-----------------|-------------------------------------------------------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Byte            | Field Description                                     | Byte            | Field Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 0               | Frame (0xFE)                                          | 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 1, 2            | Length (0x00NN)                                       | 1, 2            | Length (0x00NN)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 3, 4            | Message Type (0x0122)                                 | 3, 4            | Message Type (0x0122)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 5               | Reserved (0x00)                                       | 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 6               | Sequence Number                                       | 6               | Same Sequence Number                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 7               | Logical Node ID                                       | 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs | 8, 9            | Status (MSB, LSB)<br>0x0001 Invalid TLV Data<br>Software can't find the TLV Data Buffer. This can also occur if the Data for a TLV is out of range.<br>0x0002 Invalid Data Type<br>0x0003 Invalid number of TLVs<br>There are no TLVs in the message.<br>0x0004 Invalid TLV Length<br>The TLV length is different from what is expected.<br>0x0006 Invalid TLV<br>Unknown TLV<br>0x000D Mandatory TLVs missing<br>One or more mandatory TLVs are missing<br><br>Also see Common Response Status Values. |
|                 | Number of AEs to follow                               | 10              | Number of TLVs to Follow                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|                 | AE<br>0x01 Slot                                       |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| :               | Data Type<br>0x00 TLVs                                | :               | TLVs                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| :               | Number of TLVs to Follow                              | :               | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | <p>TLVs</p> <p>Mandatory TLV:</p> <p>0x05FA Card Object*</p> <ul style="list-style-type: none"><li>0x0000 - General Card Configuration</li><li>0x0001 - Conferencing Parameters</li><li>0x0003 - Alarm Configuration</li><li>0x0004 - DSP Series 2 Fax Parameters Query</li><li>0x0005 - Echo Cancel</li><li>0x0006 - Pvd/AMD Parameters</li><li>0x0007 - Dial Pulse Detection Parameters</li></ul> <p>Optional TLVs:</p> <ul style="list-style-type: none"><li>0x05DC File Management Configuration (mandatory for initial configuration, optional after that)</li><li>0x05DD Server Address (mandatory for initial configuration of NFS, optional after that)</li><li>0x05DF Vocabulary Index File (mandatory for initial configuration of NFS, optional after that)</li><li>0x05EF Alarm Threshold Configure (up to 16)</li><li>0x05F0 DSP 2 Main Board Memory Alarm Threshold Configure (up to 2)</li><li>0x05F1 DSP Temporary Memory Alarm Threshold Configure (up to 2)</li><li>0x05F4 Statistics Update Timer</li><li>0x0600 NFS User Group ID</li><li>0x0601 NFS Poll Retries</li><li>0x0619 PVD Parameters</li><li>0x0620 AMD Parameters</li><li>0x061B AMD Reports</li><li>0x0640 Matrix/Host Sequence Number Size*</li><li>0x0750 Dial Pulse Detection Parameters - This TLV is sent to configure the Dial Pulse Detection Parameters (0x0007). See 0x05FA Card Object TLV above.</li></ul> <p>* Use mandatory Card Object TLV 0x05FA (with Object ID 0x0000) and optional Matrix/Host Sequence Number Size TLV 0x0640 to select an 8 or 16 bit Sequence Number as follows:</p> <ul style="list-style-type: none"><li>0x00 - 8 bit</li><li>0x01 - 16 bit</li></ul> <p><u>The following optional TLVs set the card-level defaults:</u></p> <ul style="list-style-type: none"><li>0x0607 Output Gain Control (default 0db)</li><li>0x0608 Noise Gating Enable (default disabled)</li><li>0x0609 Noise Gate Parameters</li><li>0x060A Echo Suppression Enable (default disabled)</li><li>0x060B Echo Suppression Parameters</li><li>0x060C Automatic Gain Control Enable (default disabled)</li><li>0x060D Automatic Gain Control Input Level</li><li>0x060E Automatic Gain Control Parameters</li><li>0x060F Conference Failure Behavior</li></ul> <p><u>0x0000 - General Card Configuration</u></p> <p><u>0x0001 - Conferencing Parameters</u></p> <p><u>0x0003 - Alarm Configuration</u></p> |
|---|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Generic Card Configure 0x0122

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   | <u>0x0005 - Echo Cancel</u><br>0x0673 Echo Cancel Tap Length<br>0x0674 Echo Cancel NLP Type<br>0x0675 Echo Cancel ADAPT<br>0x0676 Echo Cancel Bypass<br>0x0677 Echo Cancel G.176 Modem Answer Detection<br>0x0678 Echo Cancel NLP Threshold<br>0x0679 Echo Cancel CNG Noise Threshold                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|   | <u>0x0004 - DSP Series 2 Fax Parameters Query</u><br>0x0641 Header Parameter Format<br>0x0642 T.30 Control Parameter Max Value<br>0x0643 T.30 Control Parameter Transmit Level<br>0x0644 T.30 Control Parameter ECM Enabled<br>0x0645 T.30 Control Parameter Local Session ID<br>0x0648 Receive Resolution Type<br>0x0648 Receive Resolution Type<br>0x0649 Receive Encoding Type<br>0x064B Receive Bad Line<br>0x064C Receive Page Size<br>0x0651 Receive Enable ECM<br>0x0651 Receive Enable ECM<br>0x0652 Receive Add Header<br>0x0654 Receive Line Error Threshold<br>0x0655 Receive Timeout<br>0x0657 Receive Terminal ID<br>0x065A Transmit Modem Type<br>0x065B Transmit Resolution Type<br>0x065C Transmit Page Size<br>0x0661 Transmit Enable ECM<br>0x0662 Transmit Add Header<br>0x0664 Transmit Enable CNG<br>0x0666 Transmit Timeout<br>0x0668 Transmit dbm Level<br>0x066A Transmit Terminal ID<br>0x066C Fax Processing Events Set Register |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

## Generic Card Query 0x0123

---

**SwitchKit Name** GenericCardQuery

**Type** EXS API and SwitchKit API message

**Description** **Generic Card Query 0x0123**

Use this message to query various card configuration settings.

**NOTE:** If you send too many TLVs in one message and exceed the allowable message length, you receive a Memory Allocation Failure NACK.

### DSP Series 2 Card

For the DSP Series 2 card, use this message to query the following:

- Card-level Defaults
- Alarm settings
- NFS location and Vocabulary Index File
- Sequence Number Size (8-bit or 16-bit)
- Advanced Conferencing Features
- T.30 Fax Parameters
- Echo Cancel Parameters
- Dial Pulse Detection Parameters

**Sent by** Host Application

**Related API Message** *Generic Card Configure*

**SwitchKit Code** **C Structure**

```
typedef struct {
 UBYTE Slot;
 UBYTE reserved18[29];
 UBYTE DataType;
 UBYTE TLVCount;
 UBYTE Data[221];
} XL_GenericCardQuery;
```

### C Structure Response

```
typedef struct {
 unsigned short Status;
 UBYTE reserved6[13];
 UBYTE TLVCount;
 UBYTE Data[250];
} XL_GenericCardQueryAck;
```

### C++ Class

```
class XLC_GenericCardQuery : public XLC_OutboundMessage {
public:
```

```

UBYTE getSlot() const;
void setSlot(UBYTE x);
UBYTE getDataType() const;
void setDataType(UBYTE x);
UBYTE getTLVCount() const;
void setTLVCount(UBYTE x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x)
};

```

### C++ Class Response

```

class XLC_GenericCardQueryAck : public
 XLC_OutboundMessage {
public:

 unsigned short getStatus() const
 void setStatus(unsigned short x)
 UBYTE getTLVCount() const;
 void setTLVCount(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x)
 ;

```

### EXS API Hex Format

| MESSAGE (White) |                       | RESPONSE (Gray) |                       |
|-----------------|-----------------------|-----------------|-----------------------|
| Byte            | Field Description     | Byte            | Field Description     |
| 0               | Frame (0xFE)          | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)       | 1, 2            | Length (0x00NN)       |
| 3, 4            | Message Type (0x0123) | 3, 4            | Message Type (0x0123) |
| 5               | Reserved (0x00)       | 5               | Reserved (0x00)       |
| 6               | Sequence Number       | 6               | Same Sequence Number  |
| 7               | Logical Node ID       | 7               | Logical Node ID       |

Generic Card Query 0x0123

|   |                                                                                                                 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---|-----------------------------------------------------------------------------------------------------------------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8 | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs<br><hr/> Number of AEs to follow<br><hr/> AE<br>0x01 Slot | 8, 9 | Status (MSB, LSB)<br>0x0001 Invalid TLV Data<br>Software cannot find the TLV Data Buffer.<br>This can also occur if the Data for a TLV<br>is out of range.<br>0x0002 Invalid Data Type<br>0x0003 Invalid number of TLVs<br>There are no TLVs in the message.<br>0x0004 Invalid TLV Length<br>The TLV length is different from what is<br>expected.<br>0x0006 Invalid TLV<br>Unknown TLV<br>0x000D Mandatory TLVs missing<br>One or more mandatory TLVs are missing<br><br>Also see Common Response Status Values. |
| : | Data Type<br>0x00 TLVs                                                                                          | 10   | Number of TLVs to Follow                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| : | Number of TLVs to Follow                                                                                        |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>TLVs<br/> <u>Mandatory</u><br/>                 0x05FA Card Object<br/>                 (must be the first TLV in the Message)<br/>                 0x0000 - General Card Configuration<br/>                 0x0001 - Conferencing Parameters<br/>                 0x0003 - Alarm Configuration<br/>                 0x0004 - DSP Series 2 Fax Parameters Query<br/>                 0x0005 - Echo Cancel<br/>                 0x0006 - PVD/AMD Parameters<br/>                 0x0007 - Dial Pulse Detection Parameters</p> <p>Other optional TLVs depend on the Card Object queried.</p> <p><u>0x0000 - General Card Configuration</u><br/>                 0x0640 Matrix/Host Sequence Number Size</p> <p><u>0x0001 - Conferencing Parameters</u><br/>                 0x060D Automatic Gain Control Input Level<br/>                 0x060F Conference Failure Behavior</p> <p><u>0x0003 - Alarm Configuration</u><br/>                 0x0617 Alarm Threshold Query</p> <p><u>0x0004 - DSP Series 2 Fax Parameters Query</u><br/>                 To query multiple parameters use the following:<br/>                 0x066D Fax Configuration Query Type<br/>                 0x0001 - T.30 control parameters<br/>                 0x0002 - Receive parameters<br/>                 0x0004 - Transmit parameters</p> <p>To query individual Fax parameters:<br/>                 0x0641 Header Parameter Format<br/>                 0x0642 T.30 Control Parameter Max Value<br/>                 0x0643 T.30 Control Parameter Transmit Level<br/>                 0x0644 T.30 Control Parameter ECM Enabled<br/>                 0x0645 T.30 Control Parameter Local Session ID<br/>                 0x0648 Receive Resolution Type<br/>                 0x0649 Receive Encoding Type<br/>                 0x064B Receive Bad Line<br/>                 0x064C Receive Page Size<br/>                 0x0651 Receive Enable ECM<br/>                 0x0652 Receive Add Header<br/>                 0x0654 Receive Line Error Threshold<br/>                 0x0655 Receive Timeout<br/>                 0x0657 Receive Terminal ID<br/>                 0x065A Transmit Modem Type<br/>                 0x065B Transmit Resolution Type<br/>                 0x065C Transmit Page Size<br/>                 0x0661 Transmit Enable ECM<br/>                 0x0662 Transmit Add Header</p> | <p>11</p> <p>TLVs<br/> <u>Mandatory</u><br/>                 0x05FA Card Object (always the first TLV in the Response)</p> <p>Other TLVs returned depend on the Card Object queried, as follows:</p> <p><u>0x0000 - General Card Configuration</u><br/>                 0x05DC File Management Configuration<br/>                 0x05DD Server Address<br/>                 0x05DF Vocabulary Index File<br/>                 0x05F3 Vocabulary Index Read Information<br/>                 0x0600 NFS User Group ID<br/>                 0x0601 NFS Poll Retries<br/>                 0x061C Server State</p> <p><u>0x0001 - Conferencing Parameters</u><br/>                 0x0607 Output Gain Control (Default is 0db)<br/>                 0x0608 Noise Gating Enable (default is Disabled)<br/>                 0x0609 Noise Gate Parameters<br/>                 0x060A Echo Suppression Enable<br/>                 0x060B Echo Suppression Parameters<br/>                 0x060C Automatic Gain Control Enable<br/>                 0x060E Automatic Gain Control Parameters</p> <p><u>0x0003 - Alarm Configuration</u><br/>                 0x0618 Alarm Threshold Enabled Report<br/>                 0x05EF Alarm Threshold Configure</p> <p><u>0x0004 - DSP Series 2 Fax Parameters Query</u><br/>                 For 0x066D Fax Configuration Query Type TLV<br/>                 Query Type: 0x0001 - T.30 control parameters<br/>                 0x0642 T.30 Control Parameter Max Value<br/>                 0x0643 T.30 Control Parameter Transmit Level<br/>                 0x0644 T.30 Control Parameter ECM Enabled</p> <p>Query Type: 0x0002 - Receive parameters<br/>                 0x0648 Receive Resolution Type<br/>                 0x0649 Receive Encoding Type<br/>                 0x064B Receive Bad Line<br/>                 0x064C Receive Page Size<br/>                 0x0651 Receive Enable ECM<br/>                 0x0652 Receive Add Header<br/>                 0x0654 Receive Line Error Threshold<br/>                 0x0655 Receive Timeout</p> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Generic Card Query 0x0123

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   | <p>0x0664 Transmit Enable CNG<br/>         0x0666 Transmit Timeout<br/>         0x0668 Transmit dbm Level<br/>         0x066A Transmit Terminal ID<br/>         0x066C Fax Processing Events Set Register</p> <p><u>0x0005 - Echo Cancel</u><br/>         0x0673 Echo Cancel Tap Length<br/>         0x0674 Echo Cancel NLP Type<br/>         0x0675 Echo Cancel ADAPT<br/>         0x0676 Echo Cancel Bypass<br/>         0x0677 Echo Cancel G.176 Modem Answer<br/>         Detection<br/>         0x0678 Echo Cancel NLP Threshold<br/>         0x0679 Echo Cancel CNG Noise Threshold</p> <p><u>0x0006 - PVD/AMD Parameters</u><br/>         0x0619 PVD Parameters<br/>         0x0620 AMD Parameters</p> <p><u>0x0007 - Dial Pulse Detection Parameters</u><br/>         0x05FA Card Object</p> |   | <p>Query Type: 0x0004 - Transmit parameters<br/>         0x065A Transmit Modem Type<br/>         0x065B Transmit Resolution Type<br/>         0x065C Transmit Page Size<br/>         0x0661 Transmit Enable ECM<br/>         0x0662 Transmit Add Header<br/>         0x0664 Transmit Enable CNG<br/>         0x0666 Transmit Timeout<br/>         0x0668 Transmit dbm Level</p> <p>For other Individual Parameter TLVs:<br/>         Same as sent.</p> <p><u>0x0005 - Echo Cancel</u><br/>         Same as sent.</p> <p><u>0x0006 - PVD/AMD Parameters</u><br/>         Same as sent.</p> |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

## Generic Event Logger Configure/Query 0x003D

---

**SwitchKit Name** GELConfigQuery

**Type** EXS API and SwitchKit API message

**Description** **Generic Event Logger Configure/Query 0x003D**

The Generic Event Logger (GEL) records system application and software debug events in the CSP. The GEL provides more debug data than the current fault log and online diagnostics.

This message lets you control the Generic Event Logger (GEL). Every card that supports GEL has two buffers. One buffer stores LOGF/LOGB entries, and the other stores LOGT entries.

LOGF enters seven data fields of ULONG type and an 80 character descriptor field to the LOGF/LOGB buffer. All data fields are user defined.

LOGB puts a memory dump from a user defined address into the LOGF/LOGB buffer. The user also defines the size of the memory dump and a descriptive string of up to 20 characters.

LOGT places a user-defined 20 character descriptive string into the LOGT buffer.

**NOTE:** This message is for debugging purposes only and should not be used until you contact technical support. Also, this message reduces system performance. **Dialogic recommends using this message in test systems only.**

**Sent by** Host

**SwitchKit Code** **Configuration**

```
GELConfigQuery (
 Node = integer,
 Slot = integer,
 Action = integer,
 DataLength = integer,
 Data = byte array);
```

**C Structure**

```
typedef struct {
 UBYTE Slot;
 UBYTE Action;
 UBYTE DataLength;
 UBYTE Data[221];
} XL_GELConfigQuery;
```

**C Structure Response**

```
typedef struct {
 unsigned short Status;
 UBYTE Slot;
```

```
 UBYTE Data[221];
} XL_GELConfigQueryAck;
```

### **C++ Class**

```
class XLC_GELConfigQuery : public XLC_OutboundMessage {
public:
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
 UBYTE getAction() const;
 void setAction(UBYTE x);
 UBYTE getDataLength() const;
 void setDataLength(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

### **C++ Class Response**

```
class XLC_GELConfigQueryAck : public
 XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | RESPONSE (Gray) |                                                       |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-------------------------------------------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Byte            | Field Description                                     |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 0               | Frame (0xFE)                                          |
| 1, 2            | Length (0x00NN)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 1, 2            | Length (0x00NN)                                       |
| 3, 4            | Message Type (0x003D)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 3, 4            | Message Type (0x003D)                                 |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 5               | Reserved (0x00)                                       |
| 6               | Sequence Number                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 6               | Same Sequence Number                                  |
| 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 7               | Logical Node ID                                       |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 8, 9            | Status (MSB, LSB)<br>See information below this table |
|                 | Number of AEs to follow                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                 |                                                       |
|                 | AE<br>0x01 Slot                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                 |                                                       |
| :               | <b>Action</b><br>0x01 Set control mask<br>0x02 Set Buffer Full option<br>0x03 Start logging*<br>0x04 Pause logging*<br>0x05 Clear log<br>0x06 Get logger status<br>0x07 Get control mask<br>0x08 Get Event buffer data<br>0x09 Get Trace buffer data<br>0x0A Get Module names<br>0x0B Stop Write<br>0x0C Telnet IP Print enable<br>0x0D Telnet IP Print disable<br><br>*The Start (0x03) and Pause logging (0x04) actions have a global effect. That is, if the AIBs Slot Number field is 0xFF, the actions start or pause all card Generic Event loggers. | :               | Same as message                                       |
| :               | Data[0] Data Length Field                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | :               | Response Data[0]<br>(See table below)                 |
| :               | Data[1] Data fields<br>(see table below)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | :               | Checksum                                              |
| :               | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                 |                                                       |

**Response Status Continued Below**

If the message contains invalid fields, the CSP returns a NACK, but the following are some specific reasons for NACKs to this message. If you try to configure the logger or retrieve the buffer before pausing the logger, then the CSP responds with a NACK. This NACK specifies that the logger is not in a valid state.

The following NACKs are used:

- 0x14 Message Invalid for Current CSP 2000 Matrix Card State  
The CSP Matrix Series 3 Card is not in a state to accept this message.
- 0x74 Invalid Card Type  
Incorrect card type for attempted operation.
- 0x20 Invalid Action Value  
Invalid decision parameter action value was received.
- 0x61 Invalid Slot  
The slot specified is not valid for the message sent.
- 0x99 Invalid Configuration Request  
Configuration request is invalid.
- 0x17 Invalid Data Type  
Returned in response to a message with invalid data parameters.

| Data                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Response Data                                                                                                                                                                                         |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Below is the data associated with each of the above actions.</p> <p><u>0x01 Set control mask</u><br/>Use this Action to define the control masks for each module.</p> <p>Data Length: n<br/>Data[0] Module ID 1<br/>    Bit Mask 4 bytes:<br/>    0<br/>    1<br/>    2<br/>    3<br/>    :<br/>Data[n] Module ID n<br/>    Bit Mask 4 bytes:<br/>    0<br/>    1<br/>    2<br/>    3</p>                                                                                                                                                                            | <p><u>0x01 For Set control mask</u><br/>Data[0] Paused Mask [4 bytes] for module ID 1<br/>Data[1] Paused Mask [4 bytes] for module ID 2<br/>:<br/>Data[31] Paused Mask [4 bytes] for module ID 32</p> |
| <p><u>0x02 Set Buffer Full option</u><br/>Use this Action to set up the Buffer Full option. When you use the Stop option, the buffer stops recording information once it is full. The Wrap option, once the buffer is full, the buffer begins to overwrite the earliest data.</p> <p>Data Length: 0x01<br/>Data: 0x00 Stop (stops the buffer when it is full)<br/>      0x01 Wrap (wraps to beginning, overwriting oldest data when buffer is full)</p>                                                                                                                 |                                                                                                                                                                                                       |
| <p><u>0x03 Start logging*</u><br/>Use this Action to set up the buffer to turn on after a reset or not. If you want the buffer to start logging after a reset, send this Action with a data of 0x01. If you do not want the buffer to turn on after a reset, send the data 0x00.</p> <p>Data Length: 0x01<br/>Data: 0x00 Stop (Do not continue logging to buffer after reset)<br/>      0x01 Start (Continue logging to buffer after reset)</p> <p>*If the slot number in the AIB is FF, then this message starts the loggers on all system cards that support GEL.</p> |                                                                                                                                                                                                       |

| Data                                                                                                                                                                                                                                                                                                      | Response Data                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><u>0x04 Pause logging*</u><br/>Use this message to stop logging to a buffer. Send this message again to unpause logging to a buffer.</p> <p>Data Length: 0x00<br/>Data: NA</p> <p>*If the slot number in the AIB is FF, then this message pauses the loggers on all system cards that support GEL.</p> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <p><u>0x05 Clear log</u><br/>Use this Action when you want to remove all entries from a log.</p> <p>Data Length: 0x00<br/>Data: NA</p>                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <p><u>0x06 Get logger status</u><br/>Use this Action to receive a report on the status of a buffer.</p> <p>Data Length: 0x00<br/>Data: NA</p>                                                                                                                                                             | <p><u>0x06 For Get logger status</u></p> <p>Data[0] Number of Events in buffer<br/>Data[1] Number of Blocks in buffer<br/>Data[2] Number of Trace records in buffer<br/>Data[3] Number of Records skipped<br/>Data[4] Buffer Full action<br/>    0 stop logging when full<br/>    1 wrap buffer<br/>Data[5] Logger state<br/>    0 Paused<br/>    1 Logging<br/>Data[6] Buffer state<br/>    0 Available<br/>    1 Full<br/>Data[7] Percent of LOGF/LOGB buffer remaining<br/>Data[8] Max. LOGF/LOGB buffer size (MSB)<br/>Data[9] Max. LOGF/LOGB buffer size (LSB)<br/>Data[10] Maximum LOGT records (MSB)<br/>Data[11] Maximum LOGT records (LSB)</p> |

| Data                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Response Data                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><u>0x07 Get control mask</u><br/>                     Use this Action to find out the control masks of a buffer.</p> <p>Data Length: 0x00<br/>                     Data: NA</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | <p><u>0x07 For Get control mask</u></p> <p>Data[0] Module ID 1 (MSB)<br/>                     Data[1] Module ID 1 (LSB)<br/>                     Bit Mask [paused] 4 bytes<br/>                     0<br/>                     1<br/>                     2<br/>                     3</p> <p>Bit Mask [enabled] 4 bytes<br/>                     0<br/>                     1<br/>                     2<br/>                     3</p> <p>:</p> <p>Data[n] Module ID n (MSB)<br/>                     Data[n+1] Module ID n (LSB)<br/>                     Bit Mask [paused] 4 bytes<br/>                     0<br/>                     1<br/>                     2<br/>                     3</p> <p>Mask [enabled] 4 bytes<br/>                     0<br/>                     1<br/>                     2<br/>                     3</p> |
| <p><u>0x08 Get Event buffer data</u><br/>                     Send this message to retrieve the information in the Event buffer and have it dumped to the file specified. Use Data[0] to tell the computer to either append the file you give and put the new information at the end, or to overwrite the existing file. To stop the dump, you need to send the Stop Write (0x0B) Action.</p> <p>Data Length: n<br/>                     Data[0] Write permission<br/>                     0 append<br/>                     1 overwrite<br/>                     Data[1] First character [Path/Filename]<br/>                     :<br/>                     Data[n] NULL termination</p> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

| Data                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Response Data                                                                                                                                                                                                                                           |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><u>0x09 Get Trace buffer data</u><br/>                     Send this message to retrieve the information in the Trace buffer and have it dumped to the file specified. Use Data[0] to tell the computer to either append the file you give and put the new information at the end, or to overwrite the existing file. To stop the dump, you need to send the Stop Write (0x0B) Action.</p> <p>Data Length: n<br/>                     Data:<br/>                     Data[0] Write permission<br/>                             0 append<br/>                             1 overwrite<br/>                     Data[1] First character[Path/Filename]<br/>                             :<br/>                     Data[n] NULL termination</p> |                                                                                                                                                                                                                                                         |
| <p><u>0x0A Get Module Names</u><br/>                     Use this Action to find out the names of a module.</p> <p>Data Length: 0x00<br/>                     Data: NA</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | <p><u>0x0A For Get Module Names</u></p> <p>Data[0] Module name 1 [12 characters]<br/>                     Data[1] Module name 2 [12 characters]<br/>                             :<br/>                     Data[31] Module name 32 [12 characters]</p> |
| <p><u>0x0B Stop Write</u><br/>                     Use this Action to stop writing from a buffer to a file.</p> <p>Data Length: n<br/>                     Data[0] First character [Path/Filename]<br/>                             :<br/>                     Data[n] NULL termination</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                         |

## GenerateLogMsg

---

|                    |                                                                                                                                                                                        |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Type</b>        | EXS SwitchKit API message                                                                                                                                                              |
| <b>Description</b> | This message is used by the functions <i>sk_logMessage()</i> and <i>sk_logMessageOnConnection()</i> to log a message at specified level.<br><br>This message does not have a response. |
| <b>Sent by</b>     | Function <i>sk_logMessage()</i>                                                                                                                                                        |
| <b>C Structure</b> | <pre>typedef struct {<br/>    } <i>SK_GenerateLogMsg</i>;</pre>                                                                                                                        |
| <b>C++ Class</b>   | <pre>class <i>SKC_GenerateLogMsg</i> : public SKC_AdminMessage {<br/>public:<br/>    };</pre>                                                                                          |

# Generic Report 0x0046

---

**SwitchKit Name** GenericReport

**Type** EXS API and SwitchKit API message

**Description** **Generic Report 0x0046**

This message is generated in response to the following Query Types sent in the *System Configuration Query* message:

- 0x04 - Active Conference IDs
- 0x05 - Channel State
- 0x08 - DS1 Trunk MIB Report
- 0x0A - Resource Usage
- 0x0B - DS3 Trunk MIB Report
- 0x10 - Detailed Conference Information
- 0x13 - Conference Speakers
- 0x14 - Conferencing Features
- 0x15 - Child Conference Information
- 0x16 - Child Conference IDs
- 0x18 - Cache File Query

**NOTE:** The AIB and Number of Resources fields apply only if the Report Type field is Resource Usage (0x0A)

**In SwitchKit**

If you want to get this report, you must register for it by calling the `sk_msgRegister` function with the parameter, `SK_RESOURCE_UTIL_REPORT`.

**Sent by** CSP

**Example Message (Socket Log Output for SwitchKit)**

In the following socket log output/example message, the *Generic Report* contains resource usage data for Slot 9.

```
00 63 00 46 00 00 ff 0a 00 01 01 01 09 04 00 26 03 01 02 00
02 48 c0 00 00 01 c0 03 00 02 63 00 00 00 00 05 00
05 9e 00 00 03 26 00 05 00 08 8c 00 00 00 04 00 01 01
00 02 08 00 00 f4 84 01 00 5c 05 03 20 00 00 f4 84 09
00 5c 05 01 00 11 02 00 05 03 00 0f 04 01 30 05 00 34
06 00 47 07 00 26 32 04 3d
```

**Switchkit Code C Structure**

```
typedef struct {
 UBYTE ReportType;
 UBYTE Data[252];
} XL_GenericReport;
```

**C++ Class**

```
class XLC_GenericReport : public XLC_InboundMessage {
public:
 UBYTE getReportType() const;
 void setReportType(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

**Overview of message** The following table provides an overview of this message. The table following it provides the detail for each byte.

| MESSAGE (White)                                                                            |                                   | RESPONSE (Gray) |                       |
|--------------------------------------------------------------------------------------------|-----------------------------------|-----------------|-----------------------|
| Byte                                                                                       | Field Description                 | Byte            | Field Description     |
| 0                                                                                          | Frame (0xFE)                      | 0               | Frame (0xFE)          |
| 1, 2                                                                                       | Length (0x00NN)                   | 1, 2            | Length (0x0005)       |
| 3, 4                                                                                       | Message Type (0x0046)             | 3, 4            | Message Type (0x0046) |
| 5                                                                                          | Reserved (0x00)                   | 5               | Reserved (0x00)       |
| 6                                                                                          | Sequence Number                   | 6               | Same Sequence Number  |
| 7                                                                                          | Logical Node ID                   | 7               | Logical Node ID       |
|                                                                                            |                                   | 8               | Checksum              |
| 8                                                                                          | Report Type                       |                 |                       |
| <b>NOTE:</b> The AIB and Number of Resources fields apply only if the Report Type is 0x0A. |                                   |                 |                       |
| :                                                                                          | AIB (Individual AEs)<br>0x01 Slot |                 |                       |
| :                                                                                          | Number of Resources               |                 |                       |
| :                                                                                          | Data [0]                          |                 |                       |
| :                                                                                          | :                                 |                 |                       |
| :                                                                                          | Checksum                          |                 |                       |

**EXS API Hex Format -  
Detailed**

| MESSAGE (White)                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                   | RESPONSE (Gray) |                       |
|-------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte                                                                                                        | Field Description                                                                                                                                                                                                                                                                                                                                                                 | Byte            | Field Description     |
| 0                                                                                                           | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                      | 0               | Frame (0xFE)          |
| 1, 2                                                                                                        | Length (0xNNNN)                                                                                                                                                                                                                                                                                                                                                                   | 1, 2            | Length (0x0005)       |
| 3, 4                                                                                                        | Message Type (0x0046)                                                                                                                                                                                                                                                                                                                                                             | 3, 4            | Message Type (0x0046) |
| 5                                                                                                           | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                   | 5               | Reserved (0x00)       |
| 6                                                                                                           | Sequence Number                                                                                                                                                                                                                                                                                                                                                                   | 6               | Same Sequence Number  |
| 7                                                                                                           | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                   | 7               | Logical Node ID       |
|                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                   | 8               | Checksum              |
| 8                                                                                                           | Report Type<br>0x04 - Active Conference IDs<br>0x05 - Channel State<br>0x08 - DS1 Trunk MIB Report<br>0x0A - Resource Usage<br>0x0B - DS3 Trunk MIB Report<br>0x10 - Detailed Conference Information<br>0x13 - Conference Speakers<br>0x14 - Conferencing Features Configuration<br>0x15 - Child Conference Information<br>0x16 - Child Conference IDs<br>0x18 - Cache File Query |                 |                       |
| <b>NOTE:</b> The AIB and Number of Resources fields apply only if the Report Type is Resource Usage (0x0A). |                                                                                                                                                                                                                                                                                                                                                                                   |                 |                       |
| :                                                                                                           | AIB<br>Address Method<br>0x00 - Individual AEs                                                                                                                                                                                                                                                                                                                                    |                 |                       |
|                                                                                                             | Number of AEs to follow                                                                                                                                                                                                                                                                                                                                                           |                 |                       |
|                                                                                                             | AE<br>0x01 Slot                                                                                                                                                                                                                                                                                                                                                                   |                 |                       |
| :                                                                                                           | Number of Resources                                                                                                                                                                                                                                                                                                                                                               |                 |                       |
| :                                                                                                           | Data<br>The data depends upon the Report Type<br><br>0x04 Active Conference IDs<br>Data[0,1] Number of conference IDs to report (up to 50 per message)<br>Data[2,3] First Conference ID<br>Data[4,5] Second Conference ID<br>: :                                                                                                                                                  |                 |                       |

0x05 Channel State

Data 0–6 is an AIB indicating the first channel to be reported:

- Data[0] Address Method (0x00) Individual AEs
- Data[1] Number of Address Elements (0x01)
- Data[2] Address Type (0x0D, Channel)
- Data[3] Data Length (0x03)
- Data[4] Logical Span ID, MSB
- Data[5] Logical Span ID, LSB
- Data[6] Channel

Data[7] Number of channels reported (1-32)

Data[8] First channel's current L4 state. (Repeated for each channel.)

The following are Layer 4 state values:

- 0x00 Out-of-Service
- 0x01 Incoming Call
- 0x02 Wait For CSA in Incoming Call State
- 0x03 In Service Idle
- 0x04 Incoming Call, Wait for L5
- 0x05 Outgoing Call, L3 Outseize Wait
- 0x06 Wait For CSA in Incoming Call Wait for Host State
- 0x07 Answered
- 0x08 L4 Clear ACK Wait, L3 Disconnect Received
- 0x09 L3 Clear Wait
- 0x0A Busied out
- 0x0B Outgoing Call Cut-thru
- 0x0C Wait for CSA in Incoming Alerted State
- 0x0D Wait for CSA in Incoming in Answered State
- 0x0E Incoming Call Alerted
- 0x0F L3 Clear Wait
- 0x10 Wait for CSA in Outgoing Cut-thru
- 0x11 Wait for CSA in Outgoing Alerted State
- 0x12 L4 Clear ACK Wait, L3 Clear Received
- 0x13 Outgoing Call Alerted
- 0x14 L4 Clear ACK Wait, L5 Clear Received
- 0x15 L5 Release Wait
- 0x16 Incoming Call, L4 Clear ACK Wait (Park Processing)
- 0x17 Incoming Call Alerted, L4 Clear ACK Wait (Park Processing)
- 0x18 Answered, L4 Clear ACK Wait (Park Processing)
- 0x19 Outgoing Alerted, L4 Clear ACK Wait (Park Processing)
- 0x1A Outgoing Cut-thru, L4 Clear ACK Wait (Park Processing)
- 0x1B Wait for CSA (Call Service Ack) for Internal Routing

Data[9] First channel's current L3 state as viewed by L4  
(Repeated for each channel.)

The following are the Layer 3 state values relevant to Layer 4:

- 0x00 Out-of-Service
- 0x01 In Service
- 0x02 Maintenance
- 0x03 In Service Blocked
- 0x04 Unblocked

: :

0x08 - DS1 Trunk MIB Report

Data 0-5 is an AIB:

- Data[0] Address Method (0x00)
- Data[1] Number of Address Elements (0x01)
- Data[2] Address Type (0x0C)
- Data[3] Data Length (0x02)
- Data[4-5] Logical Span
- Data [6-9] Errored seconds
- Data [10-13] Severly errored seconds
- Data [14-17] Severly errored framing seconds
- Data [18-21] Unavailable seconds
- Data [22-25] Controlled slip seconds
- Data [26-29] Path code violations
- Data [30-33] Line code violations
- Data [34-37] Line errored seconds
- Data [38-41] Bursty errored seconds
- Data [42-45] Degraded minutes
- Data [46-49] Time elapsed
- Data [50-53] Interval number

0x0A Resource Usage

Resource ID 0x00 Memory

- Data[0] Length
- Data[1] Number of regions configured on the card
- Data[2] Number of partitions configured on the card
- Data[3] Region ID of Nth region
- Data[4-7] Size of the region in bytes
- Data[8-11] Number of bytes used from the region
- Data [12] Partition ID of Nth partition
- Data[13-16] Size of the partition in bytes
- Data[17-20] Used bytes of partition
- :
- Data[:] Region ID of Nth region(1 byte)
- Data[:] Size of the Nth region in bytes (4 bytes)
- Data[:] Number of bytes used from the Nth region(4 bytes)
- Data [:] Partition ID of 1st partition (1 byte)
- Data[:] Size of the 1st partition in bytes (4 bytes)
- Data[:] Used bytes of 1st partition (4 bytes)
- :
- Data [:] Partition ID of Nth partition (1 byte)
- Data[:] Size of the Nth partition in bytes (4 bytes)
- Data[:] Used bytes of Nth partition (4 bytes)

Resource ID 0x01 MCB

- Data[0] Length
- Data[0] Message buffer usage

Resource ID 0x02 CPU Usage Level 1

- Data[0] Length
- Data[1-4] Time in milliseconds covering this report
- Data[5] Number of tasks reported
- Data[6] Logical task ID (idle task only 0x00)
- Data[7] Percentage of time usage for task (integer part)
- Data[8] Percentage of time usage for task (decimal part)

|                                                                         |                                                                                                                                        |
|-------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| Resource ID 0x03 CPU Usage Level 2                                      |                                                                                                                                        |
| Data[0]                                                                 | Length                                                                                                                                 |
| Data[1-4]                                                               | Time in milliseconds covering this report                                                                                              |
| Data[5]                                                                 | Number of tasks reported                                                                                                               |
| Data[6]                                                                 | Logical task ID                                                                                                                        |
| Data[7]                                                                 | Percentage of time usage for task (integer part)                                                                                       |
| Data[8]                                                                 | Percentage of time usage for task (decimal part)                                                                                       |
| 0x0B DS3 Trunk MIB Report Type                                          |                                                                                                                                        |
| Data 0-5 is an AIB:                                                     |                                                                                                                                        |
| Data[0]                                                                 | Address Method (0x00)                                                                                                                  |
| Data[1]                                                                 | Number of Address Elements (0x01)                                                                                                      |
| Data[2]                                                                 | Address Type (0x32)                                                                                                                    |
| Data[3]                                                                 | Data Length (0x02)                                                                                                                     |
| Data[4]                                                                 | Slot Number                                                                                                                            |
| Data[5]                                                                 | DS3 Offset                                                                                                                             |
| Data[6]                                                                 | Alarm Status                                                                                                                           |
| A bit mask indicating what alarm is being sent and received on the DS3: |                                                                                                                                        |
| Bit 0: Receiving Red                                                    |                                                                                                                                        |
| Bit 1: Receiving RAI                                                    |                                                                                                                                        |
| Bit 2: Receiving Loss of Signal (LOS)                                   |                                                                                                                                        |
| Bit 3: Receiving Out of Frame (OOF)                                     |                                                                                                                                        |
| Bit 4: Receiving AIS                                                    |                                                                                                                                        |
| Bit 5: Sending AIS                                                      |                                                                                                                                        |
| Bit 6: Sending RAI                                                      |                                                                                                                                        |
| Bit 7: Reserved                                                         |                                                                                                                                        |
| Data[7-10]                                                              | Code Violations – Near End – Line (CV-L)                                                                                               |
| Data[11-14]                                                             | Errored Seconds – Near End – Line (ES-L)<br>Seconds containing one or more BPVs or EXZs defects.                                       |
| Data[15-18]                                                             | Severely Errored Seconds – Near End - Line (SES-L)<br>Seconds during which BPVs plus EXZs exceed 44.                                   |
| Data[19-22]                                                             | Code Violations – Near End – Path (CVP-P)                                                                                              |
| Data[23-26]                                                             | Errored Seconds – Near End – Path (ESP-P)<br>Seconds containing one or more p-bit parity errors or SEF defects                         |
| Data[27-30]                                                             | Severely Errored Seconds – Near End -Path (SESP-P)<br>Seconds during which p-bit parity errors exceed 44 or one or more SEF defects    |
| Data[31-34]                                                             | Unavailable Seconds – Near End - Path (UASP-P)                                                                                         |
| Data[35-38]                                                             | Code Violations – Near End - Path (CVCP-P)                                                                                             |
| Data[39-42]                                                             | Errored Seconds – Near End – Path (ESCP-P)<br>Seconds containing one or more c-bit parity errors or SEF defects.                       |
| Data[43-46]                                                             | Severely Errored Seconds – Near End - Path (SESCP-P)<br>Seconds during which c-bit parity errors exceed 44 or one or more SEF defects. |
| Data[47-50]                                                             | Unavailable Seconds – Near End - Path (UASCP-P)                                                                                        |
| Data[51-54]                                                             | SEF/AIS Seconds – Near End –Path (SAS-P)<br>Seconds containing one or more SEF defects.                                                |
| Data[55-58]                                                             | Code Violations – Far End – Path (CVCP-PFE)                                                                                            |
| Data[59-62]                                                             | Errored Seconds – Far End – Path (ESCP-PFE)                                                                                            |
| Data[63-66]                                                             | Severely Errored Seconds – Far End - Path (SESCP-PFE)                                                                                  |
| Data[67-70]                                                             | Unavailable Seconds – Far End – Path (UASCP-PFE)                                                                                       |
| Data[71-74]                                                             | SEF/AIS Seconds – Far End – Path (SASCP-PFE)                                                                                           |
| Data[75-78]                                                             | Time Elapsed since the counters were last cleared                                                                                      |
| Data[79-82]                                                             | Interval Number (number of reports sent including this one)                                                                            |

|               |                                                                                                                                                                 |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x10          | Detailed Conference Information                                                                                                                                 |
| Data[0,1]     | Conference ID                                                                                                                                                   |
| Data[2]       | Number of Channel Block's                                                                                                                                       |
| Data[3]       | Block Number                                                                                                                                                    |
| Data[4]       | Node ID                                                                                                                                                         |
| Data[5]       | DSP Card Slot                                                                                                                                                   |
| Data[6]       | Module ID                                                                                                                                                       |
| Data[7]       | DSP Number                                                                                                                                                      |
| Data[8]       | Conference Number                                                                                                                                               |
| Data[9]       | Conference Type                                                                                                                                                 |
| Data[10]      | Broadcast Enabled                                                                                                                                               |
| Data[11, 12]  | Number of Channels Allocated                                                                                                                                    |
| Data[13,14]   | Total Number of participant's                                                                                                                                   |
| Data[15]      | Tone Attached                                                                                                                                                   |
| Data[16,17]   | Tone ID                                                                                                                                                         |
| Data[18]      | VRA Attached                                                                                                                                                    |
| Data[19]      | Marked for Deletion                                                                                                                                             |
| Data[20]      | Recording                                                                                                                                                       |
| Data[21-32]   | Spare Bytes                                                                                                                                                     |
| Data[33,34]   | Number of Participant's in block                                                                                                                                |
| Data[35,36]   | Logical Span for Participant Number 1                                                                                                                           |
| Data[37]      | Logical Channel for Participant Number 1                                                                                                                        |
| Data[38]      | Leg value is 1 for 1-Way Listen-Only Connected Channels<br>Leg value is 2 for 2-Way Connected Channels<br>Leg value is 3 for 1-Way Talk-Only Connected Channels |
| Data[39,40]   | Logical Span for Participant Number 2                                                                                                                           |
| Data[41]      | Logical Channel for Participant Number 2                                                                                                                        |
| Data[42]      | 1 or 2 legs for Participant Number 2                                                                                                                            |
| Data[... ]    | Continue with a list of span's and channels                                                                                                                     |
| Data[159,160] | Logical Span for Participant Number 32                                                                                                                          |
| Data[161]     | Logical Channel for Participant Number 32                                                                                                                       |
| Data[162]     | 1 or 2 legs for Participant Number 32                                                                                                                           |
| 0x13          | Conference Speakers (Parent Conference)                                                                                                                         |
| Data[0-5]     | Conference ID AIB                                                                                                                                               |
| Data[6, 7]    | Speaker #1 Span                                                                                                                                                 |
| Data[8]       | Speaker #1 Channel                                                                                                                                              |
| Data[9]       | Speaker #1 Average Signal Magnitude                                                                                                                             |
| Data[10-13]   | Speaker #1 Active Speaking Time (5 ms increments)                                                                                                               |
| Data[14, 15]  | Speaker #2 Span                                                                                                                                                 |
| Data[16]      | Speaker #2 Channel                                                                                                                                              |
| Data[17]      | Speaker #2 Average Signal Magnitude                                                                                                                             |
| Data[18-21]   | Speaker #2 Active Speaking Time (5 ms increments)                                                                                                               |
| Data[22, 23]  | Speaker #3 Span                                                                                                                                                 |
| Data[24]      | Speaker #3 Channel                                                                                                                                              |
| Data[25]      | Speaker #3 Average Signal Magnitude                                                                                                                             |
| Data[26-29]   | Speaker #3 Active Speaking Time (5 ms increments)                                                                                                               |

0x13 Conference Speakers (Child Conference)

If conference speaker information is queried for child conference, the report data will be in the following format.

Data[0-7] Child Conference ID AIB  
Data[8-9] Speaker #1 Span  
Data[10] Speaker #1 Chan  
Data[11] Speaker #1 Energy Level (dBm)  
Data[12-15] Speaker #1 Active Speaking Time  
Data[16-17] Speaker #2 Span  
Data[18] Speaker #2 Chan  
Data[19] Speaker #1 Energy Level (dBm)  
Data[20-23] Speaker #2 Active Speaking Time  
Data[24-25] Speaker #3 Span  
Data[26] Speaker #3 Chan  
Data[27] Speaker #1 Energy Level (dBm)  
Data[28-31] Speaker #3 Active Speaking Time

0x14 Conferencing Features (Parent Conference)

If a conference ID is queried, the Report data is in the following format:

Data[0-5] Conference ID AIB  
Data[6-7] DTMF Clamping Enabled  
Data[8-9] Output Gain  
Data[10-11] Noise Gating Enabled  
Data[12-13] Noise Gating Time Constant  
Data[14-15] Max Noise Level  
Data[16-17] Noise Gating Sensitivity  
Data[18-19] Echo Suppression Enable  
Data[20-21] Echo Return Loss  
Data[22-23] AGC Enable  
Data[24-25] AGC Input Level  
Data[26-27] AGC Time Constant  
Data[28-29] Conference Failure Behavior  
Data[30-31] Input Gain  
Data [32-33] Transit Connection Mode

If a channel is queried the Report data is in the following format:

Data[0-6] Channel AIB  
Data[7-8] DTMF Clamping Enabled  
Data[9-10] Output Gain  
Data[11-12] Noise Gating Enabled  
Data[13-14] Noise Gating Time Constant  
Data[15-16] Max Noise Level  
Data[17-18] Noise Gating Sensitivity  
Data[19-20] Echo Suppression Enable  
Data[21-22] Echo Return Loss  
Data[23-24] AGC Enable  
Data[25-26] AGC Input Level  
Data[27-28] AGC Time Constant  
Data[29-30] Conference Failure Behavior  
Data[31-32] Channel Input Gain

0x14 Conferencing Features (Child Conference)

If parameters are queried for child conference, the report data will be in the following format.

Data[0-7] Child Conference ID AIB  
Data[8-9] DTMF Clamping Enabled  
Data[10-11] Output Gain  
Data[12-13] Noise Gating Enabled  
Data[14-15] Noise Gating Time Constant  
Data[16-17] Max Noise Level  
Data[18-19] Noise Gating Sensitivity  
Data[20-21] Echo Suppression Enable  
Data[22-23] Echo Return Loss  
Data[24-25] AGC Enable  
Data[26-27] AGC Input Level  
Data[28-29] AGC Time Constant  
Data[30-31] Conference Failure Behavior  
Data [32-33] Input Gain  
Data [34-35] Transit Connection Mode

If parameters are queried for a particular channel connected to a conference/child conference, the report data will be in the following format.

Data[0-6] Channel AIB  
Data[7-8] DTMF Clamping Enabled  
Data[9-10] Output Gain  
Data[11-12] Noise Gating Enabled  
Data[13-14] Noise Gating Time Constant  
Data[15-16] Max Noise Level  
Data[17-18] Noise Gating Sensitivity  
Data[19-20] Echo Suppression Enable  
Data[21-22] Echo Return Loss  
Data[23-24] AGC Enable  
Data[25-26] AGC Input Level  
Data[27-28] AGC Time Constant  
Data[29-30] Conference Failure Behavior  
Data[31-32] Channel Input Gain

| 0x15 Detailed Child Conference Information |                                  |
|--------------------------------------------|----------------------------------|
| Data[0-1]                                  | Parent Conference ID             |
| Data[2-3]                                  | Child Conference ID              |
| Data[4]                                    | Number of Channel Blocks         |
| Data[5]                                    | Block Number                     |
| Data[6]                                    | Node ID                          |
| Data[7]                                    | DSP Card Slot                    |
| Data[8]                                    | Module ID                        |
| Data[9]                                    | DSP Number                       |
| Data[10]                                   | Conference Number                |
| Data[11]                                   | Conference Type                  |
| Data[12]                                   | Broadcast Enabled                |
| Data[13-14]                                | Number of Channels Allocated     |
| Data[15-16]                                | Total number of participants     |
| Data[17]                                   | Tone attached                    |
| Data[18,19]                                | Tone ID                          |
| Data[20]                                   | VRA Attached                     |
| Data[21]                                   | Marked for Deletion              |
| Data[22]                                   | Recording                        |
| Data[23-34]                                | Spare Bytes                      |
| Data[35-36]                                | Number of Participants in Block  |
| Data[37-38]                                | Logical Span for Participant #1  |
| Data[39]                                   | Logical Chan for Participant #1  |
| Data[40]                                   | 1 or 2 legs for Participant #1   |
| .                                          | .                                |
| .                                          | .                                |
| .                                          | .                                |
| Data[161-162]                              | Logical Span for Participant #32 |
| Data[163]                                  | Logical Chan for Participant #32 |
| Data[164]                                  | 1 or 2 legs for Participant #32  |

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   | <p>(0x16) Child Conference IDs Information Report type</p> <p>Data[0-5] Conference ID AIB</p> <p>Data[6-7] Number of child conferences Ids to report (Up to 50 per message)</p> <p>Data[8-9] First Child Conference ID</p> <p>Data[10-11] Second Child Conference ID</p> <p>Data[12-13] Third Child Conference ID</p> <p style="text-align: center;">.</p> <p>Data[106-107] Fiftieth Child Conference ID</p><br><p>0x18 Cache File Query Report type</p> <p>If the Cache File ID information is queried, the report data will be in the following format.</p> <p>Data[0] Slot Number</p> <p>The following data is returned to the DSP Series 2 card main board cache and for each DSP chip.</p> <p>Data[1] DSP Module Number<br/>0x00, 0x01, 0xFF</p> <p>Data[2] DSP Chip Number<br/>0x00, 0x01, 0x02, 0x03, 0xFF</p> <p>Data[3-6] File ID</p> <p>Data[7] File Status<br/>0x00 - Absent<br/>0x01 - Present</p> <p>Data[8] Encoding Format<br/>0x00 - A-law<br/>0x01 - Mu-law</p> <p>Data[9] File Format<br/>0x00 - raw (packed raw binary data)<br/>0x01 - vox<br/>0x02 - wav</p> <p>Data[10-13] File Size (in bytes)</p> <p>Note: Data[8-13] is valid only if Data[7] File Status is set to 0x01</p> |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

# GenericLLCReport

---

- Type** SwitchKit API message
- Description** This message reports the status of LLC entities to applications that have registered for it. In particular it will report the status of the LLC socket connections, LLC to LLC redundancy status, and the service state of channels groups being watched by an application.
- Sent by** LLC to applications that have registered for it using the SK\_msgRegister() function with the parameter SK\_GENERIC\_LLC\_REPORT.
- C Structure**
- ```
typedef struct {
    unsigned short DataSize;
    UBYTE ReportType;
    UBYTE TLVCount;
    UBYTE Data[249];
} SK_GenericLLCReport;
```
- C++ Class**
- ```
class SKC_GenericLLCReport : public SKC_ToolkitMessage
{
public:
 unsigned short getDataSize() const;
 void setDataSize(unsigned short x);
 UBYTE getReportType() const;
 void setReportType(UBYTE x);
 UBYTE getTLVCount() const;
 void setTLVCount(UBYTE x) ;
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```
- Arguments** The following table shows the arguments sent by the LLC:

| Argument   | Description                  |
|------------|------------------------------|
| DataSize   | Number of bytes of the data. |
| ReportType | Type of GenericLLCReport     |
| TLVCount   | Number of TLVs in the data.  |
| data       | Data bytes                   |

## Report Types

The following report types are used in the *GenericLLCReport* message.

| Report Type             | Value                     |
|-------------------------|---------------------------|
| SK_NSR_CHANGE           | 0x01 (not currently used) |
| SK_NSQ_CHANGE           | 0x02 (not currently used) |
| SK_PM_CHANGE            | 0x03                      |
| SK_SOCKET_CHANGE        | 0x04 (not currently used) |
| SK_SSC_UPDATE           | 0x05                      |
| SK_LINK_UP              | 0x06                      |
| SK_LINK_DOWN            | 0x07                      |
| SK_GROUP_WATCHER_STATUS | 0x08                      |
| SK_CHANNEL_RANGE_STATUS | 0x09                      |
| SK_LLC_REDUNDANCY       | 0x0b                      |

**TLVs** Several different TLVs can be reported within the *GenericLLCReport*:

- SK\_LLCREDSTATE\_TLV (0x0004)
- SK\_LLCREDSTATE\_TLV (0x0005)
- SK\_LLCREDSETTINGS\_TLV (0x0006)
- SK\_POLL\_DATA\_TLV (0x1000)
- Sk\_IPV4\_TLV (0x1001)
- SK\_ROUTING\_TLV (0x1002)
- Sk\_LNI\_TLV (0x1003)
- SK\_CHANNEL\_GROUP\_TLV (0x1004)
- SK\_CHANNEL\_RANGE\_TLV (0x1005)

### Report Type: SK\_LLC\_REDUNDANCY

A GenericLLCReport of report type, SK\_LLC\_REDUNDANCY is generated and sent to registered applications by the active LLC when:

- An application connects
- The LLC changes redundancy states
- An *LLCQuery* of type SK\_LLC\_REDUNDANCY\_QUERY\_TLV is sent to the active LLC.

The standby LLC sends reports under the same circumstances. However, because only the active LLC can communicate with the

standby LLC, the primary LLC (PLLC) to redundant LLC (RLLC) link must be in SK\_LINK\_UP state for the application to receive the reports. If the PLLC to RLLC link is in state SK\_LINK\_DOWN, no responses from the standby LLC will be seen.

### Possible TLV Combinations

| Report Type       | LLCREDTYPE_TLV | LLCREDSTATE_TLV | LLCREDSETTINGS_TLV |
|-------------------|----------------|-----------------|--------------------|
| SK_LLC_Redundancy | 1              | 1               | 1                  |

**TLV Syntax** The TLVs described in this section are used with the SK\_LLC\_REDUNDANCY report type in a *GenericLLCReport* message.

- SK\_LLCREDTYPE\_TLV (0x0004)
- SK\_LLCREDSTATE\_TLV (0x0005)
- SK\_LLCREDSETTINGS\_TLV (0x0006)

**SK\_LLCCREDTYPE\_TLV (0x0004)**

| Description | Byte (Value)                                                                                             |
|-------------|----------------------------------------------------------------------------------------------------------|
| Tag         | Data[0, 1] = 0x0004 (SK_LLCCREDTYPE_TLV)                                                                 |
| Length      | Data[2, 3] = 0x0001                                                                                      |
| Value       | Data[4] = Type<br><br>0x01 - LLC was started as primary<br>0x02 - LLC was started as redundant (with -r) |

**SK\_LLCCREDSTATE\_TLV (0x0005)**

| Description | Byte (Value)                                                                                                                                                                                                                                                            |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tag         | Data[0, 1] = 0x0005 (SK_LLCCREDSTATE_TLV)                                                                                                                                                                                                                               |
| Length      | Data[2, 3] = 0x0002                                                                                                                                                                                                                                                     |
| Value       | Data[4] = LLC state<br><br>0x00 - Transient or Unknown<br>0x01 - Active<br>0x02 - Standby<br><br>Data [5] = LLC to LLC Redundant Link State<br><br>0x06 - RLink Up (PLLC to RLLC communication possible)<br>0x07 - Rlink Down (PLLC to RLLC communication not possible) |

**SK\_LLCCREDSETTINGS\_TLV (0x006)**

| Description | Byte (Value)                                    |
|-------------|-------------------------------------------------|
| Tag         | Data[0, 1] = 0x0006<br>(SK_LLCCREDSETTINGS_TLV) |
| Length      | Data[2, 3] = 0x0012                             |

| Description | Byte (Value)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Value       | <p><b>IP Address of the Host this message came from (presumably the Active LLC)</b><br/> Data [4] = First octet of Host IP address<br/> Data [5] = Second octet of Host IP address<br/> Data [6] = Third octet of Host IP address<br/> Data [7] = Fourth octet of Host IP address</p> <p><b>IP of the Primary LLC's listening socket</b><br/> Data [8] = First octet of PLLC IP address<br/> Data [9] = Second octet of PLLC Host IP address<br/> Data [10] = Third octet of PLLC Host IP address<br/> Data [11] = Fourth octet of PLLC IP address</p> <p><b>Listening Port of the Primary LLC</b><br/> Data [12, 13] = PLLC Port</p> <p><b>IP of the Redundant LLC's listening socket</b><br/> Data [14] = First octet of RLLC IP address<br/> Data [15] = Second octet of RLLC Host IP address<br/> Data [16] = Third octet of RLLC Host IP address<br/> Data [17] = Fourth octet of RLLC IP address</p> <p><b>Listening Port of the Redundant LLC</b><br/> Data [18, 19] = RLLC</p> <p><b>PortSwitchback mode as defined by Primary Host</b><br/> Data [20] = mode<br/> 0x00 - Primary LLC preferred<br/> 0x01 - Manual</p> |

### Report Types

The TLVs in this section are used to report the status of the LLC socket connections.

- SK\_PM\_CHANGE
- SK\_SSC\_UPDATE
- SK\_LINK\_UP
- SK\_LINK\_DOWN

These report types are unsolicited messages that report the status of socket connections between the LLC and its devices, in particular, devices that share a logical node but are on unique logical links (i.e. direct connections to the SS7 for TCAP).

The following list describes the conditions that will cause the LLC to send these reports within the GenericLLCReport:

- **SK\_PM\_CHANGE**  
Sent when the first poll is received by the LLC for a CSP Matrix Series 3 Card. Also sent any time a value changes in the Poll. The UpdatedFieldBits field indicates which fields have changed since the last poll.
- **SK\_SSC\_UPDATE**  
Sent when an application issues an AddLLCCard. An SSC\_UPDATE may be generated if the LLC already knew about the device and the LLC is currently connected to the active device within that node. In other words, if the Link is considered to be up.
- **SK\_LINK\_UP**  
Sent to indicate that there is a connection to the active device for this node and the device is ready to be configured. This will be sent if this is a new condition (i.e. the LLC was not previously connected to the active Device).
- **SK\_LINK\_DOWN**  
Sent to indicate that there is NO connection to the active Device for a node. This will be sent if this is a new condition (that is, we were previously connected to the active Device).

**Important!** Performing an AddLLCCard will result in an SK\_SSC\_UPDATE with the last poll from the active Device, if the LLC is currently connected to the Device in the active state.

**SK\_POLL\_DATA\_TLV (0x1000)**

| <b>Description</b> | <b>Byte (Value)</b>                                                                                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tag                | Data[0, 1] = 0x1000 (SK_POLL_DATA_TLV)                                                                                                                                                                                          |
| Length             | Data[2, 3] = 0x000b                                                                                                                                                                                                             |
| Value              | Data[4] = StatusData<br>Data[5,6] = SystemType<br>Data[7] = MatrixSide<br>Data[8] = MatrixState<br>Data[9] = AdjMatrixState<br>Data[10] = StatusBits<br>Data[11,12] = Actual Logical Node ID<br>Data[13, 14] = UpdatedFieldBits |

**SK\_IPV4\_TLV (0x1001)**

| <b>Description</b> | <b>Byte (Value)</b>                                                                                                                                                                                                |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tag                | Data[0, 1] = 0x1001 (SK_IPV4_TLV)                                                                                                                                                                                  |
| Length             | Data[2, 3] = 0x0004                                                                                                                                                                                                |
| Value              | Data[4] = First octet of IP address (hexadecimal)<br>Data[5] = Second octet of IP address (hexadecimal)<br>Data[6] = Third octet of IP address (hexadecimal)<br>Data[7] = Fourth octet of IP address (hexadecimal) |

**SK\_ROUTING\_TLV (0x1002)**

| <b>Description</b> | <b>Byte (Value)</b>                  |
|--------------------|--------------------------------------|
| Tag                | Data[0, 1] = 0x1002 (SK_ROUTING_TLV) |
| Length             | Data[2, 3] = 0x0004                  |
| Value              | Data[4,5,6,7] = Routing ID           |

**SK\_LNI\_TLV (0x1003)**

| <b>Description</b> | <b>Byte (Value)</b>                   |
|--------------------|---------------------------------------|
| Tag                | Data[0, 1] = 0x1003 (SK_LNI_TLV)      |
| Length             | Data[2, 3] = 0x0002                   |
| Value              | Data[4,5] = Requested Logical Node ID |

## Report Types

The TLVs in this section are used to report the service state of channel groups being watched by an application.

- SK\_GROUP\_WATCHER\_STATUS
- SK\_CHANNEL\_RANGE\_STATUS

| Report Type             | SK_CHANNEL_GROUP_TLV | SK_CHANNEL_RANGE_TLV |
|-------------------------|----------------------|----------------------|
| SK_GROUP_WATCHER_STATUS | 1                    |                      |
| SK_CHANNEL_RANGE_STATUS |                      | 1                    |

### SK\_CHANNEL\_GROUP\_TLV (0x1004)

In the event that the environment variable SK\_CHANNEL\_RECOVERY\_METHOD=0x03 (All\_OUT\_OF\_SERVICE) is set or a *ForceGroupState* message is received, the LLC has the ability to take entire groups of channels in-service and out-of-service. In the event that this occurs a "master" application can be created that monitors this behavior by registering for GenericLLCReports and parsing a new report type within it called SK\_GROUP\_WATCHER\_STATUS.

| Description | Byte (Value)                                                                                                                                    |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Tag         | Data[0, 1] = 0x1004<br>(SK_CHANNEL_GROUP_TLV)                                                                                                   |
| Length      | Data[2, 3] = 0x0034                                                                                                                             |
| Value       | Data[4] = Status<br><br>0x00 Not Watched<br>0x01 Watched<br><br>Data[5] Reserved (Not Used)<br><br>Data [6,56] = Group (null terminated string) |

### SK\_CHANNEL\_RANGE\_TLV (0x1005)

In the event that the environment variable SK\_CHANNEL\_RECOVERY\_METHOD=0x03 (All\_OUT\_OF\_SERVICE) is set or a *ForceGroupState* message is

received, the LLC has the ability to take entire groups of channels in-service and out-of-service. In the event that this occurs a "master" application can be created that monitors this behavior by registering for GenericLLCReports and parsing a new report type within it called SK\_CHANNEL\_RANGE\_STATUS.

| Description | Byte (Value)                                                                                                                                                                                                        |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tag         | Data[0, 1] = 0x1005<br>(SK_CHANNEL_RANGE_TLV)                                                                                                                                                                       |
| Length      | Data[2, 3] = 0x0009                                                                                                                                                                                                 |
| Value       | Data[4,5] = Start Span<br>Data[6] =Start Channel<br>Data[7,8] = End Span<br>Data[9] = End Channel<br>Data[10] = Status<br><br>0xF0 SK_IN_SERVICE<br>0x0F SK_OUT_OF_SERVICE<br><br>Data[11,12] = Reserved (Not Used) |

#### Possible TLV Combinations

| Report Type   | SK_LNI_TLV | SK_POLL_DATA_TLV | SK_IPV4_TLV |
|---------------|------------|------------------|-------------|
| SK_PM_CHANGE  | 1          | 1                | 1           |
| SK_SSC_UPDATE | 1          | 1                | 1           |
| SK_LINK_UP    | 1          | 0                | *           |
| SK_LINK_DOWN  | 1          | 0                | **          |

\* A Report Type will contain at least one of these TLVs. The CSP Matrix Series 3 Card represented by the first TLV in a series will always indicate the current Active.

\*\* A Report Type will contain at least one of these TLVs. When a series of TLVs is sent by the LLC, the order of the TLVs is irrelevant. When the link is down the LLC does not know which is Active or not.

# HexTool

---

|                    |                                                                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Type</b>        | SwitchKit API message                                                                                                                                                                       |
| <b>Purpose</b>     | Use the <i>SK_HexTool</i> message to send a configuration file directly to a specific slot in a node.                                                                                       |
| <b>Description</b> | This message sends a config file containing EXS API to a specific slot. The *.cfg file is resent whenever required, based on the configuration setting for the card in the designated slot. |
| <b>Sent by</b>     | CSA or SwitchManager                                                                                                                                                                        |
| <b>Arguments</b>   | The following table shows the user modifiable arguments of this message:                                                                                                                    |

| Arguments   | Descriptions                                                                                   |
|-------------|------------------------------------------------------------------------------------------------|
| Slot        | The slot number the file is destined for.                                                      |
| CfgFile     | The filename of the file to parse and send. The maximum size of the CfgFile name is 160 bytes. |
| CSAFilename | Internally used by CSA.                                                                        |

|                      |                                                                                                                                                                                                                                                                              |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Configuration</b> | <pre><i>HexTool</i> (   Node = integer,   Slot = integer,   CfgFile = quoted string   ConnectionID = integer);</pre>                                                                                                                                                         |
| <b>C Structure</b>   | <pre>typedef struct {   UBYTE Slot;   char CfgFile[100];   char CSAFilename[100]; } <i>SK_HexTool</i>;</pre>                                                                                                                                                                 |
| <b>C++ Class</b>     | <pre>class <i>SKC_HexTool</i> : public SKC_DummyMessage { public:   UBYTE getSlot() const;   void setSlot(UBYTE x);   const char *getCfgFile() const;   void setCfgFile(const char *x);   const char *getCSAFilename() const;   void setCSAFilename(const char *x); };</pre> |



# HostAlarm

---

- Type** SwitchKit API message
- Description** Use the *SK\_HostAlarm* message to allow SwitchKit applications to generate alarms or events that can be sent to other SwitchKit applications. Receiving applications, such as SNMP processes will be able to register to receive this class of message.
- Sent by** Application
- Arguments** The following table shows the arguments you can change:

| Argument       | Description                                                                                                       |
|----------------|-------------------------------------------------------------------------------------------------------------------|
| AppName        | A text string used to uniquely identify the application sending the message. (ASCII String 64 characters maximum) |
| AlarmTime      | This is the number of seconds since January 1 1970.                                                               |
| IPAddrOfSender | This field will contain the IP Address of the host system where the problem originated                            |
| AlarmMajor     | See Table 1:Alarm Types                                                                                           |
| AlarmSubType   | See Table 2: Alarm Sub-types/Probable Cause                                                                       |
| AlarmSeverity  | See Table 3:Alarm Severity                                                                                        |
| TLVCount       | The number of data items associated with this message                                                             |
| TLVData        | SeeTable 4: TLV Data Types                                                                                        |

**Configuration** Syntax, if able to send, is through SMgr

**C Structure**

```
typedef struct {
 UBYTE IPAddrOfSenderSize;
 UBYTE reserved5[12];
 char AppName[64];
 int AlarmTime;
 UBYTE IPAddrOfSender[4];
 UBYTE AlarmMajor;
```

```

 UBYTE AlarmSubType;
 UBYTE AlarmSeverity;
 UBYTE TLVCount;
 UBYTE TLVData[177];
} SK_HostAlarm;

```

**C Structure Response**

```

typedef struct {
 int Status;
 UBYTE reserved21[4];
} SK_HostAlarmAck;

```

**C++ Class**

```

class SKC_HostAlarm : public SKC_ToolkitMessage {
public:
 UBYTE getIPAddrOfSenderSize() const;
 void setIPAddrOfSenderSize(UBYTE x);
 const char *getAppName() const;
 void setAppName(const char *x);
 int getAlarmTime() const;
 void setAlarmTime(int x);
 const UBYTE *getIPAddrOfSender() const;
 UBYTE *getIPAddrOfSender();
 void setIPAddrOfSender(UBYTE *x);
 UBYTE getAlarmMajor() const;
 void setAlarmMajor(UBYTE x);
 UBYTE getAlarmSubType() const;
 void setAlarmSubType(UBYTE x);
 UBYTE getAlarmSeverity() const;
 void setAlarmSeverity(UBYTE x);
 UBYTE getTLVCount() const;
 void setTLVCount(UBYTE x);
 const UBYTE *getTLVData() const;
 UBYTE *getTLVData();
 void setTLVData(UBYTE *x);
};

```

**C++ Class Response**

```

class SKC_HostAlarmAck : public SKC_ToolkitAck {
public:
 int getStatus() const
 void setStatus(int x)
};

```

**Table 1:Alarm Types**

| Name                      | Value as Defined    | Description                                                                                                                                    |
|---------------------------|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Communications Alarm Type | SK_COMMUNICATIONS 1 | An alarm of this type is principally associated with the procedures and/or processes required to convey information from one point to another. |

| Name                        | Value as Defined      | Description                                                                                                                 |
|-----------------------------|-----------------------|-----------------------------------------------------------------------------------------------------------------------------|
| QOS Alarm Type              | SK_QUALITYOFSERVICE 2 | An alarm of this type is principally associated with degradation in the quality of a service.                               |
| Processing Error Alarm Type | SK_PROCESSINGERROR 3  | An alarm of this type is principally associated with a software or processing fault.                                        |
| Equipment Alarm Type        | SK_EQUIPMENT 4        | An alarm of this type is principally associated with an equipment fault                                                     |
| Environmental Alarm Type    | SK_ENVIRONMENTAL 5    | An alarm of this type is principally associated with a condition relating to an enclosure in which the equipment resides.it |

**Table 2: Alarm Sub-types/Probable Cause**

| Name                             | Value as Defined                   | Description                                                                                                                                                                              |
|----------------------------------|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Adapter error                    | SK_ADAPTERERROR 1                  |                                                                                                                                                                                          |
| Application subsystem failure:   | SK_APPLICATIONSUBSYSTEMFAILURE 2   | A failure in an application subsystem has occurred (an application subsystem may include software to support the Session, Presentation or Application layers)                            |
| Bandwidth reduced                | SK_BANDWIDTHREDUCED 3              | The available transmission bandwidth has decreased                                                                                                                                       |
| Call establishment error         | SK_CALLESTABLISHMENTERROR 4        | An error occurred while attempting to establish a connection                                                                                                                             |
| Communications protocol error    | SK_COMMUNICATIONPROTOCOLERROR 5    | A communication protocol has been violated                                                                                                                                               |
| Communications subsystem failure | SK_COMMUNICATIONSUBSYSTEMFAILURE 6 | A failure in a subsystem that supports communications over telecommunications links, these may be implemented via leased telephone lines, by X.25 networks, token-ring LAN, or otherwise |

| Name                                  | Value as Defined                       | Description                                                                                                                                                                                            |
|---------------------------------------|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Configuration or customization error: | SK_CONFIGURATIONORCUSTOMIZATIONERROR 7 | A system or device generation or customization parameter has been specified incorrectly, or is inconsistent with the actual configuration                                                              |
| Congestion:                           | SK_CONGESTION 8                        | A system or network component has reached its capacity or is approaching it                                                                                                                            |
| Corrupt data:                         | SK_CORRUPTDATA 9                       | An error has caused data to be incorrect and thus unreliable                                                                                                                                           |
| CPU cycles limit exceeded             | SK_CPUCYCLESLIMITEXCEEDED 10           | A Central Processing Unit has issued an unacceptable number of instructions to accomplish a task                                                                                                       |
| Dataset or modem error                | SK_DATASETORMODEMERROR 11              | An internal error has occurred on a dataset or modem                                                                                                                                                   |
| Degraded signal                       | SK_DEGRADED SIGNAL 12                  | The quality or reliability of transmitted data has decreased                                                                                                                                           |
| DTE-DCE interface error:              | SK_DTEDCEINTERFACEERROR 13             | A problem in a DTE-DCE interface, which includes the interface between the DTE and DCE, any protocol used to communicate between the DTE and DCE and information provided by the DCE about the circuit |
| Enclosure door open                   | SK_ENCLOSUREDOOROPEN 14                |                                                                                                                                                                                                        |
| Equipment malfunction                 | SK_EQUIPMENTMALFUNCTION 15             | An internal machine error has occurred for which no more specific Probable cause has been identified                                                                                                   |
| Excessive vibration:                  | SK_EXCESSIVEVIBRATION 16               | Vibratory or seismic limits have been exceeded                                                                                                                                                         |
| File error:                           | SK_FILEERROR 17                        | The format of a file (or set of files) is incorrect and thus cannot be used reliably in processing                                                                                                     |
| Fire detected                         | SK_FIREDETECTED 18                     |                                                                                                                                                                                                        |
| Flood detected                        | SK_FLOODDETECTED 19                    |                                                                                                                                                                                                        |

| <b>Name</b>                                    | <b>Value as Defined</b>              | <b>Description</b>                                                                                       |
|------------------------------------------------|--------------------------------------|----------------------------------------------------------------------------------------------------------|
| Framing error                                  | SK_FRAMINGERROR 20                   | An error in the information that delimits the bit groups within a continuous stream of bits              |
| Heating/ventilation/<br>cooling system problem | SK_HEATORVENTORCOOLSYSTEM<br>PROBLEM |                                                                                                          |
| Humidity unacceptable                          | SK_HUMIDITYUNACCEPTABLE<br>22        | The humidity is not within acceptable limits                                                             |
| I/O device error                               | SK_INPUTOUTPUTDEVICEERROR<br>23      | An error has occurred on the I/O device                                                                  |
| Input device error                             | SK_INPUTDEVICEERROR 24               | An error has occurred on the input device                                                                |
| LAN error                                      | SK_IANERROR 25                       | An error has been detected on a local area network                                                       |
| Leak detected:                                 | SK_LEAKDETECTED 26                   | A leakage of (non-toxic) fluid or gas has been detected                                                  |
| Local node transmission<br>error:              | SK_LOCALNODETRANSMISSIONER<br>ROR 27 | An error occurred on a communications channel between the local node and an adjacent node                |
| Loss of frame                                  | SK_LOSSOFFRAME 28                    | An inability to locate the information that delimits the bit grouping within a continuous stream of bits |
| Loss of signal                                 | SK_LOSSOFSIGNAL 29                   | An error condition in which no data is present on a communications circuit or channel                    |
| Material supply<br>exhausted                   | SK_MATERIALSUPPLYEXHAUSTED<br>30     | A supply of needed material has been exhausted                                                           |
| Multiplexer problem                            | SK_MULTIPLEXERPROBLEM 31             | An error has occurred while multiplexing communications signals                                          |
| Out of memory                                  | SK_OUTOFMEMORY 32                    | There is no program-addressable storage available.                                                       |
| Output device error                            | SK_OUTPUTDEVICEERROR 33              | An error has occurred on the output device                                                               |
| Performance degraded                           | SK_PERFORMANCEDEGRADED<br>34         | Service agreements or service limits are outside of acceptable limits                                    |

| <b>Name</b>                            | <b>Value as Defined</b>                      | <b>Description</b>                                                                                                         |
|----------------------------------------|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| Power problem                          | SK_POWERPROBLEM 35                           | There is a problem with the power supply for one or more resources                                                         |
| Pressure unacceptable                  | SK_PRESSUREUNACCEPTABLE 36                   | A fluid or gas pressure is not within acceptable limits                                                                    |
| Processor problem                      | SK_PROCESSORPROBLEM 37                       | An internal machine error has occurred on a Central Processing Unit                                                        |
| Pump failure                           | SK_PUMPFailure 38                            | Failure of mechanism that transports a fluid by inducing pressure differentials within the fluid                           |
| Queue size exceeded                    | SK_QUEUE SIZE EXCEEDED 39                    | The number of items to be processed (configurable or not) has exceeded the maximum allowable                               |
| Receive failure                        | SK_RECEIVE FAILED 40                         |                                                                                                                            |
| Receiver failure                       | SK_RECEIVER FAILED 41                        |                                                                                                                            |
| Remote node transmission error         | SK_REMOTE NODE TRANSMISSION ERROR 42         | An error occurred on a communication channel beyond the adjacent node                                                      |
| Resource at or nearing capacity        | SK_RESOURCE AT OR NEARING CAPACITY 43        | The usage of a resource is at or nearing the maximum allowable capacity                                                    |
| Response time excessive                | SK_RESPONSE TIME EXCESSIVE 44                | The elapsed time between the end of an inquiry and beginning of the answer to that inquiry is outside of acceptable limits |
| Retransmission rate excessive          | SK_RETRANSMISSION RATE EXCESSIVE 45          | The number of repeat transmissions is outside of acceptable limits                                                         |
| Software error                         | SK_SOFTWARE ERROR 46                         | A software error has occurred for which no more specific Probable cause can be identified                                  |
| Software program abnormally terminated | SK_SOFTWARE PROGRAM ABNORMALLY TERMINATED 47 | A software program has abnormally terminated due to some unrecoverable error condition                                     |
| Software program error                 | SK_SOFTWARE PROGRAM ERROR 48                 | An error has occurred within a software program that has caused incorrect results                                          |
| Storage capacity problem               | SK_STORAGE CAPACITY PROBLEM 49               | A storage device has very little or no space available to store additional data                                            |

| Name                            | Value as Defined                    | Description                                                                                                                                         |
|---------------------------------|-------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Temperature unacceptable        | SK_TEMPERATUREUNACCEPTABLE 50       | A temperature is not within acceptable limits                                                                                                       |
| Threshold crossed               | SK_THRESHOLDCROSSED 51              | A limit (configurable or not) has been exceeded                                                                                                     |
| Timing problem                  | SK_TIMINGPROBLEM 52                 | A process that requires timed execution and/or coordination cannot complete, or has completed but cannot be considered reliable                     |
| Toxic leak detected             | SK_TOXICLEAKDETECTED 53             | A leakage of toxic fluid or gas has been detected                                                                                                   |
| Transmit failure                | SK_TRANSMITFAILURE 54               |                                                                                                                                                     |
| Transmitter failure             | SK_TRANSMITTERFAILURE 55            |                                                                                                                                                     |
| Underlying resource unavailable | SK_UNDERLYINGRESOURCEUNAVAILABLE 56 | An entity upon which the reporting object depends has become unavailable                                                                            |
| Version mismatch                | SK_VERSIONMISMATCH 57               | There is a conflict in the functionality of versions of two or more communicating entities which may affect any processing involving those entities |
| License verification failure    | SK_LICENSE 58                       | A node's license verification failed.                                                                                                               |
| Node in use                     | SK_NODE_IN_USE 59                   | An attempt was made to add a duplicate node to the LLC's node list.                                                                                 |
| Node created                    | SK_NODE_CREATED 60                  | A node was added to the LLC's node list. Initiates a severity level of SK_INFORMING.                                                                |
| Node removed                    | SK_NODE_REMOVED 61                  | A node was removed from the LLC's node list. Initiates a severity level of SK_INFORMING                                                             |

**Table 3: Alarm Severity**

| <b>Name</b>   | <b>Value as Defined</b> | <b>Description</b>                                                                                                                                                                                                                                                                                                                                               |
|---------------|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Cleared       | SK_CLEARED 1            | The Cleared severity level indicates the clearing of one or more previously reported alarms. This alarm clears all alarms for this managed object that have the same Alarm type, Probable cause and Specific problems (if given). Multiple associated notifications may be cleared by using the Correlated notifications parameter (defined below).              |
| Indeterminate | SK_INDETERMINATE 2      | The Indeterminate severity level indicates that the severity level cannot be determined.                                                                                                                                                                                                                                                                         |
| Major         | SK_MAJOR 5              | The Major severity level indicates that a service affecting condition has developed and an urgent corrective action is required. Such a severity can be reported, for example, when there is a severe degradation in the capability of the managed object and its full capability must be restored.                                                              |
| Critical      | SK_CRITICAL 6           | The Critical severity level indicates that a service affecting condition has occurred and an immediate corrective action is required. Such a severity can be reported, for example, when a managed object becomes totally out of service and its capability must be restored.                                                                                    |
| Minor         | SK_MINOR 4              | The Minor severity level indicates the existence of a non-service affecting fault condition and that corrective action should be taken in order to prevent a more serious (for example, service affecting) fault. Such a severity can be reported, for example, when the detected alarm condition is not currently degrading the capacity of the managed object. |
| Warning       | SK_WARNING 3            | The Warning severity level indicates the detection of a potential or impending service affecting fault, before any significant effects have been felt. Action should be taken to further diagnose (if necessary) and correct the problem in order to prevent it from becoming a more serious service-affecting fault.                                            |

**Table 4: TLV Data Types**

| <b>Name</b>   | <b>Value as Defined</b> | <b>Description</b>                                                                                            |
|---------------|-------------------------|---------------------------------------------------------------------------------------------------------------|
| DisplayString | SK_TLV_DISPLAYSTRING 1  | ASCII text string. Characters greater than or equal to 0x20 and characters less than 0x7e, 64 characters max. |
| Integer       | SK_TLV_INTEGER 2        | Integer field                                                                                                 |

## HostAlarm

| <b>Name</b>     | <b>Value as Defined</b>  | <b>Description</b>                                                                                                                        |
|-----------------|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| IpAddress       | SK_TLV_IPADDRESS 3       | This application-wide type represents a 32-bit internet address. It is represented as an OCTET STRING of length 4, in network byte-order. |
| PhysicalAddress | SK_TLV_PHYSICALADDRESS 4 | This data type is used to model media addresses. For example, an ethernet address would be represented as a string of 6 octets.           |

## IgnoreChanGroup

---

|                    |                                                                                                             |
|--------------------|-------------------------------------------------------------------------------------------------------------|
| <b>Type</b>        | SwitchKit API message                                                                                       |
| <b>Description</b> | This message is used by the function <b>sk_ignoreChanGroup()</b> if an application calls for that function. |
| <b>Sent by</b>     | Function <b>sk_ignoreChanGroup</b>                                                                          |
| <b>C Structure</b> | <pre>typedef struct {<br/>    } <b>SK_IgnoreChanGroup</b>;</pre>                                            |
| <b>C++ Class</b>   | <pre>class SKC_IgnoreChanGroup : public SKC_AdminMessage {<br/>public:<br/>    };</pre>                     |

# Inpulsing Parameters Configure 0x0028

---

**SwitchKit Name**    InpulsingParametersConfig

**Type**            EXS API and SwitchKit API message

**Description**    **Inpulsing Parameters Configure 0x0028**

The *Inpulsing Parameters Configure* message allows the host to configure impulse data collection parameters on specified channels. Use impulse parameters when data is collected by means of an *Inseize Control* instruction. Call setup inpulsing parameters define the address signaling type, number of digit strings, and collection method used during call setup.

The CSP supports four different inpulsing stages, and each stage is defined as either one or two digit strings. When instructing the CSP to collect address signaling information that is typically presented with in-band dual frequency tones, the host must specify a preprogrammed inpulsing stage that describes how to perform the digit collection.

Each channel has four inpulsing stages that can be preprogrammed. The inpulsing stage configuration options include: the address signaling type (DTMF, MFR1, MFR2), the number of strings (1 or 2), and the string collection method (fixed number digits, KP/ST framed, compelled).

**Sent by**            Host

**Example Message (Socket Log Output for SwitchKit)**

The following socket log output/example message shows data being sent from the host to the CSP to configure impulse data collection parameters on span 0x00, channels 0x00 - 0x17. The stage number being configured is 0x01 and the number of digit strings in the stage is 0x01. The stage complete timeout is three seconds and the address signaling type is DTMF. This message collects seven digits.

```
00 1a 00 28 00 00 ff 01 02 0d 03 00 00 00 0d 03 00 00 17 01 01 01 2c 01 01
07 01 07
```

**SwitchKit Code**    **Configuration**

```
InpulsingParametersConfig (
 Node = integer,
 Range = StartSpan:StartChan - EndSpan:EndChan,
 Stage = integer,
 NumDigitStrings = integer,
 StageCompleteTimeout = integer,
 AddrSignallingType = integer,
 String1Method = integer,
 String1Data = integer,
 String2Method = integer,
 String2Data = integer);
```

```

C Structure typedef struct {
 unsigned short StartSpan;
 UBYTE StartChannel;
 unsigned short EndSpan;
 UBYTE EndChannel;
 UBYTE Stage;
 UBYTE NumDigitStrings;
 unsigned short StageCompleteTimeout;
 UBYTE AddrSignallingType;
 UBYTE String1Method;
 UBYTE String1Data;
 UBYTE String2Method;
 UBYTE String2Data;
 } XL_InpulsingParametersConfig;

```

### **C++ Class**

```

class XLC_InpulsingParametersConfig : public
 XLC_ChanRangeMessage {
public:
 unsigned short getStartSpan() const;
 void setStartSpan(unsigned short x);
 UBYTE getStartChannel() const;
 void setStartChannel(UBYTE x);
 unsigned short getEndSpan() const;
 void setEndSpan(unsigned short x);
 UBYTE getEndChannel() const;
 void setEndChannel(UBYTE x);
 UBYTE getStage() const;
 void setStage(UBYTE x);
 UBYTE getNumDigitStrings() const;
 void setNumDigitStrings(UBYTE x);
 unsigned short getStageCompleteTimeout() const;
 void setStageCompleteTimeout(unsigned short x);
 UBYTE getAddrSignallingType() const;
 void setAddrSignallingType(UBYTE x);
 UBYTE getString1Method() const;
 void setString1Method(UBYTE x);
 UBYTE getString1Data() const;
 void setString1Data(UBYTE x);
 UBYTE getString2Method() const;
 void setString2Method(UBYTE x);
 UBYTE getString2Data() const;
 void setString2Data(UBYTE x);
};

```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                        | RESPONSE (Gray) |                       |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                                      | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                           | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)                                                                                                                                                                                                                                                                                        | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x0028)                                                                                                                                                                                                                                                                                  | 3, 4            | Message Type (0x0028) |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                        | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                                                                                                                                                                        | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                                                                                                                                                                        | 7               | Logical Node ID       |
| :               | <u>AIB</u><br>Address Method<br>0x01 - Range of AEs                                                                                                                                                                                                                                                    | 8, 9            | Status (MSB, LSB)     |
|                 | Number of AEs to follow                                                                                                                                                                                                                                                                                | 10              | Checksum              |
|                 | AEs<br>0x0D Channel (Starting)<br>0x0D Channel (Ending)                                                                                                                                                                                                                                                |                 |                       |
|                 |                                                                                                                                                                                                                                                                                                        |                 |                       |
| :               | Stage Number<br>Inpulsing Stage number to configure (1–4)                                                                                                                                                                                                                                              |                 |                       |
| :               | Number of Digit Strings<br>Number of digit strings in stage (1–2)                                                                                                                                                                                                                                      |                 |                       |
| :               | Stage Complete Time Out (MSB, LSB)<br>The maximum amount of time from the detection of the first digit until all digits have been collected for the stage. The first digit must be detected or the receive first digit timer must expire before this timer is active. See the “Timer Conversion Table” |                 |                       |
|                 | Address Signaling Type<br>The address signaling type must be the same for both strings within a stage, but may differ between stages.<br>0x01 DTMF<br>0x02 MFR1<br>0x03 MFR2<br>0x04 Reserved<br>0x05 Dial Pulse                                                                                       |                 |                       |

Impulsing Parameters Configure 0x0028

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   | <p>String 1: Collection Method</p> <p>0x01 Fixed Number of Digits</p> <p>0x02 Use Termination Digit</p> <p>0x03 Use KP/ST Signals; Do Not Send To Host<br/>         KP and ST signals (ST, STI, STII, STIII) are not included in the string sent to the host in the <i>Request For Service</i> message. This method is only valid for MFR1 signaling.</p> <p>0x04 Reserved</p> <p>0x05 Compelled<br/>         This method is only valid for MFR2 signaling.</p> <p>0x06 Use KP/ST Signals; Include in Digit String<br/>         KP and ST signals (ST, STI, STII, STIII) will be included in the string sent to the host in the <i>Request For Service</i> message. This method is only valid for MFR1 signaling.</p> <p>0x07 Compelled KP (MF-MDR1)<br/>         The KP and ST digits will not be included in the string that is reported to the host.</p> <p>0x08 Fixed or Indefinite Number of Digits<br/>         If the first digit detected is zero, the Indefinite Number of Digits method is used and the Completion Timer or Interdigit Timer is used (whichever expires first) to terminate digit collection. If the first digit collected is not zero, the Fixed Number of Digits method is used. This method is used to help facilitate collecting digit strings on channels that process national and international calls.</p> |
|   | <p>String 1: Collection Data</p> <p>Data is dependent on the value of the <i>String Collection Method</i>.</p> <p>Fixed Number of Digits<br/>         Digit Count (0x00–0x64).<br/>         Set this value to 0xFF if you want the CSP to continue requesting Forward R2 signals until no more are received. This option is valid for MFR2 only.</p> <p>Use Termination Digit<br/>         Digit Value (0x00–0x0F)</p> <p>Use KP_ST Signals<br/>         No data (0x00)</p> <p>Compelled<br/>         Digit Count (0x00–0x64).<br/>         Set this value to 0xFF if you want the CSP to continue requesting Forward R2 signals until no more are received. This option is valid for MFR2 only.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|   | <p>String 2: Collection Method</p> <p>If only one string is being collected, enter 0x00 for this field.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|   | <p>String 2: Collection Data</p> <p>If only one string is being collected, enter 0x00 for this field.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**Default Inpulse Parameters** The number of strings for each stage for each span listed below is 1.

**T1 Spans**

The information for each stage is identical.

Stages 1–4

|                        |                                                    |
|------------------------|----------------------------------------------------|
| Stage Complete Timer   | 12,000 ms                                          |
| Address Signaling Type | MFR1                                               |
| Collection Method      | Use KP_ST, KP, and ST digits not reported to host. |

**E1 Spans**

Stage 1

|                        |                  |
|------------------------|------------------|
| Stage Complete Timer   | 12,000 ms        |
| Address Signaling Type | MFR2             |
| Collection Method      | Compelled        |
| Collection Data        | 0x03 (Area Code) |

Stage 2

|                        |                             |
|------------------------|-----------------------------|
| Stage Complete Timer   | 12,000 ms                   |
| Address Signaling Type | MFR2                        |
| Collection Method      | Compelled                   |
| Collection Data        | 0x0A (CLI, Calling Line ID) |

Stage 3

|                        |                             |
|------------------------|-----------------------------|
| Stage Complete Timer   | 12,000 ms                   |
| Address Signaling Type | MFR2                        |
| Collection Method      | Compelled                   |
| Collection Data        | 0x07 (Called Party Address) |

Stage 4

|                        |                               |
|------------------------|-------------------------------|
| Stage Complete Timer   | 12,000 ms                     |
| Address Signaling Type | MFR2                          |
| Collection Method      | Compelled                     |
| Collection Data        | 0x01 (Group 2 Category Digit) |



# Inpulsing Parameters Query 0x0089

---

**SwitchKit Name** InpulsingParametersQuery

**Type** EXS API and SwitchKit API message

**Description** **Inpulsing Parameters Query 0x0089**

This host uses this message to query the inpulsing parameters configured on the specified channel. The response from the CSP reports four two-string stages, even if there is no configuration for a stage (which is reported as 0x00).

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
} XL_InpulsingParametersQuery;
```

**C Structure Response**

```
typedef struct {
 unsigned short Status;
 unsigned short Stage1Timer;
 UBYTE Stage1Strings;
 UBYTE Reserved1;
 UBYTE Stage1String1Type;
 UBYTE Stage1String1Method;
 UBYTE Stage1String1MethodData;
 UBYTE Reserved11;
 UBYTE Stage1String2Type;
 UBYTE Stage1String2Method;
 UBYTE Stage1String2MethodData;
 UBYTE Reserved12;
 unsigned short Stage2Timer;
 UBYTE Stage2Strings;
 UBYTE Reserved2;
 UBYTE Stage2String1Type;
 UBYTE Stage2String1Method;
 UBYTE Stage2String1MethodData;
 UBYTE Reserved21;
 UBYTE Stage2String2Type;
 UBYTE Stage2String2Method;
 UBYTE Stage2String2MethodData;
 UBYTE Reserved22;
 unsigned short Stage3Timer;
 UBYTE Stage3Strings;
 UBYTE Reserved3;
 UBYTE Stage3String1Type;
 UBYTE Stage3String1Method;
 UBYTE Stage3String1MethodData;
 UBYTE Reserved31;
```

```

 UBYTE Stage3String2Type;
 UBYTE Stage3String2Method;
 UBYTE Stage3String2MethodData;
 UBYTE Reserved32;
 unsigned short Stage4Timer;
 UBYTE Stage4Strings;
 UBYTE Reserved4;
 UBYTE Stage4String1Type;
 UBYTE Stage4String1Method;
 UBYTE Stage4String1MethodData;
 UBYTE Reserved41;
 UBYTE Stage4String2Type;
 UBYTE Stage4String2Method;
 UBYTE Stage4String2MethodData;
 UBYTE Reserved42;
} XL_InpulsingParametersQueryAck;

```

### **C++ Class**

```

class XLC_InpulsingParametersQuery : public
 XLC_OneChannelOutbound {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
};

```

### **C++ Class Response**

```

class XLC_InpulsingParametersQueryAck : public
 XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 unsigned short getStage1Timer() const;
 void setStage1Timer(unsigned short x);
 UBYTE getStage1Strings() const;
 void setStage1Strings(UBYTE x);
 UBYTE getReserved1() const;
 void setReserved1(UBYTE x);
 UBYTE getStage1String1Type() const;
 void setStage1String1Type(UBYTE x);
 UBYTE getStage1String1Method() const;
 void setStage1String1Method(UBYTE x);
 UBYTE getStage1String1MethodData() const;
 void setStage1String1MethodData(UBYTE x);
 UBYTE getReserved11() const;
 void setReserved11(UBYTE x);
 UBYTE getStage1String2Type() const;
 void setStage1String2Type(UBYTE x);
 UBYTE getStage1String2Method() const;
 void setStage1String2Method(UBYTE x);
 UBYTE getStage1String2MethodData() const;
 void setStage1String2MethodData(UBYTE x);
 UBYTE getReserved12() const;
 void setReserved12(UBYTE x);
 unsigned short getStage2Timer() const;
 void setStage2Timer(unsigned short x);
};

```

```

UBYTE getStage2Strings() const;
void setStage2Strings(UBYTE x);
UBYTE getReserved2() const;
void setReserved2(UBYTE x);
UBYTE getStage2String1Type() const;
void setStage2String1Type(UBYTE x);
UBYTE getStage2String1Method() const;
void setStage2String1Method(UBYTE x);
UBYTE getStage2String1MethodData() const;
void setStage2String1MethodData(UBYTE x);
UBYTE getReserved21() const;
void setReserved21(UBYTE x);
UBYTE getStage2String2Type() const;
void setStage2String2Type(UBYTE x);
UBYTE getStage2String2Method() const;
void setStage2String2Method(UBYTE x);
UBYTE getStage2String2MethodData() const;
void setStage2String2MethodData(UBYTE x);
UBYTE getReserved22() const;
void setReserved22(UBYTE x);
unsigned short getStage3Timer() const;
void setStage3Timer(unsigned short x);
UBYTE getStage3Strings() const;
void setStage3Strings(UBYTE x);
UBYTE getReserved3() const;
void setReserved3(UBYTE x);
UBYTE getStage3String1Type() const;
void setStage3String1Type(UBYTE x);
UBYTE getStage3String1Method() const;
void setStage3String1Method(UBYTE x);
UBYTE getStage3String1MethodData() const;
void setStage3String1MethodData(UBYTE x);
UBYTE getReserved31() const;
void setReserved31(UBYTE x);
UBYTE getStage3String2Type() const;
void setStage3String2Type(UBYTE x);
UBYTE getStage3String2Method() const;
void setStage3String2Method(UBYTE x);
UBYTE getStage3String2MethodData() const;
void setStage3String2MethodData(UBYTE x);
UBYTE getReserved32() const;
void setReserved32(UBYTE x);
unsigned short getStage4Timer() const;
void setStage4Timer(unsigned short x);
UBYTE getStage4Strings() const;
void setStage4Strings(UBYTE x);
UBYTE getReserved4() const;
void setReserved4(UBYTE x);
UBYTE getStage4String1Type() const ;
void setStage4String1Type(UBYTE x);
UBYTE getStage4String1Method() const;
void setStage4String1Method(UBYTE x);
UBYTE getStage4String1MethodData() const;
void setStage4String1MethodData(UBYTE x);
UBYTE getReserved41() const;
void setReserved41(UBYTE x);
UBYTE getStage4String2Type() const;
void setStage4String2Type(UBYTE x);
UBYTE getStage4String2Method() const;

```

## Inpulsing Parameters Query 0x0089

```

void setStage4String2Method(UBYTE x);
UBYTE getStage4String2MethodData() const;
void setStage4String2MethodData(UBYTE x);
UBYTE getReserved42() const;
void setReserved42(UBYTE x);
};

```

### EXS API Hex Format

| MESSAGE |                                                                                                        | RESPONSE |                       |
|---------|--------------------------------------------------------------------------------------------------------|----------|-----------------------|
| Byte    | Field Description                                                                                      | Byte     | Field Description     |
| 0       | Frame (0xFE)                                                                                           | 0        | Frame (0xFE)          |
| 1, 2    | Length (0x0004)                                                                                        | 1, 2     | Length (0x0037)       |
| 3, 4    | Message Type (0x0089)                                                                                  | 3, 4     | Message Type (0x0089) |
| 5       | Reserved (0x00)                                                                                        | 5        | Reserved (0x00)       |
| 6       | Sequence Number                                                                                        | 6        | Same Sequence Number  |
| 7       | Logical Node ID                                                                                        | 7        | Logical Node ID       |
| :       | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs<br>Number of AEs to follow<br>AE<br>0x0D Channel | 8, 9     | Status MSB, ISB       |
| :       | Checksum                                                                                               |          |                       |

| RESPONSE continued below. |         |                           |
|---------------------------|---------|---------------------------|
| 10-13                     | Stage 1 | Stage Complete Timer, MSB |
|                           |         | Stage Complete Timer, LSB |
|                           |         | Number of Digit Strings   |
|                           |         | Reserved (0x00)           |

Inpulsing Parameters Query 0x0089

|       |          |                                                                                                                                                                                                                                                                                                                                                                       |
|-------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14-17 | String 1 | Address Signaling Type<br>0x01 DTMF<br>0x02 MFR1<br>0x03 MFR2<br>0x05 Dial Pulse<br>NOTE: These values are identical for each field labeled "Address Signaling Type"                                                                                                                                                                                                  |
|       |          | Collection Method<br>0x01 Fixed Number of Digits<br>0x02 Use Termination Digit<br>0x03 Use KP_ST Signals, KP and ST digits not reported to host.<br>0x05 Compelled<br>0x06 Use KP digit and any ST digit, KP, and ST digits are reported to host.<br>NOTE: These values are identical for each field labeled "Collection Method"                                      |
|       |          | String Collection Data<br>The value of the <i>String Collection Data</i> field depends on the value of the <i>Collection Method</i> field.<br><br>Fixed Number of Digits: Digit Count<br>Use Termination Digit: Digit Value<br>Use KP_ST 0x0B (ST digit)<br>Compelled Digit Count<br>NOTE: These values are identical for each field labeled "String Collection Data" |
|       |          | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                       |
| 18-21 | String 2 | Address Signaling Type                                                                                                                                                                                                                                                                                                                                                |
|       |          | Collection Method                                                                                                                                                                                                                                                                                                                                                     |
|       |          | String Collection Data                                                                                                                                                                                                                                                                                                                                                |
|       |          | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                       |
| 22-25 | Stage 2  | Stage Complete Timer, MSB                                                                                                                                                                                                                                                                                                                                             |
|       |          | Stage Complete Timer, LSB                                                                                                                                                                                                                                                                                                                                             |
|       |          | Number of Digit Strings                                                                                                                                                                                                                                                                                                                                               |
|       |          | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                       |
| 26-29 | String 1 | Address Signaling Type                                                                                                                                                                                                                                                                                                                                                |
|       |          | Collection Method                                                                                                                                                                                                                                                                                                                                                     |
|       |          | String Collection Data                                                                                                                                                                                                                                                                                                                                                |
|       |          | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                       |
| 30-33 | String 2 | Address Signaling Type                                                                                                                                                                                                                                                                                                                                                |
|       |          | Collection Method                                                                                                                                                                                                                                                                                                                                                     |
|       |          | String Collection Data                                                                                                                                                                                                                                                                                                                                                |
|       |          | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                       |
| 34-37 | Stage 3  | Stage Complete Timer, MSB                                                                                                                                                                                                                                                                                                                                             |
|       |          | Stage Complete Timer, LSB                                                                                                                                                                                                                                                                                                                                             |
|       |          | Number of Digit Strings                                                                                                                                                                                                                                                                                                                                               |
|       |          | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                       |

Inpulsing Parameters Query 0x0089

|       |          |                           |
|-------|----------|---------------------------|
| 38-41 | String 1 | Address Signaling Type    |
|       |          | Collection Method         |
|       |          | String Collection Data    |
|       |          | Reserved (0x00)           |
| 42-45 | String 2 | Address Signaling Type    |
|       |          | Collection Method         |
|       |          | String Collection Data    |
|       |          | Reserved (0x00)           |
| 46-49 | Stage 4  | Stage Complete Timer, MSB |
|       |          | Stage Complete Timer, LSB |
|       |          | Number of Digit Strings   |
|       |          | Reserved (0x00)           |
| 50-53 | String 1 | Address Signaling Type    |
|       |          | Collection Method         |
|       |          | String Collection Data    |
|       |          | Reserved (0x00)           |
| 54-57 | String 2 | Address Signaling Type    |
|       |          | Collection Method         |
|       |          | String Collection Data    |
|       |          | Reserved (0x00)           |
| 58    | Checksum |                           |

# Inseize Control 0x002B

---

**SwitchKit Name** InseizeControl

**Type** EXS API and SwitchKit API message

**Description** **Inseize Control 0x002B**

Use this message to control incoming call setup during real-time call processing on a specified channel. Instructions and data are passed in this message in the form of Information Control Blocks (ICBs).

You can preprogram a list of instructions using the *Inseize Instruction List Configure* message and initiate them by sending the *Inseize Control* message with an ICB of “Use Instruction List”.

The system does not replace the default instruction list with the list you specify with the “Use Instruction List” ICB. The system concatenates the list you specify in this message to the default list.

An instruction that is not carried out is reported via a negative acknowledgment. A positive acknowledgment is reported only if you include a “Send Host Acknowledgment” Action ICB in the message.

**Related Messages**

- *Inseize Instruction List Configure* 0x0029 (InseizeInstrListConfig)
- *Call Control Instructions Query* 0x0087 (CallControlInstructionQuery)

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
 UBYTE ICBCount;
 UBYTE ICBDData[222];
} XL_InseizeControl;
```

**C++ Class**

```
class XL_InseizeControl : public XLC_OneChannelOutbound
{
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 UBYTE getICBCount() const;
 void setICBCount(UBYTE x);
 const UBYTE *getICBDData() const;
 UBYTE *getICBDData();
 void setICBDData(UBYTE *x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | RESPONSE (Gray) |                       |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x002B)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 3, 4            | Message Type (0x002B) |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 7               | Logical Node ID       |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs<br>Number of AEs to follow<br>AE<br>0x0D Channel                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 8, 9            | Status (MSB, LSB)     |
| :               | Number of ICBs to follow (Ignore this field if no ICBs in message.)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 10              | Checksum              |
| :               | ICB<br>Valid ICBs for this message are shown below.<br><br>0x01 Action<br>0x00 Null *<br>0x01 Report Call Processing Event<br>0x02 Generate Call Processing Event<br>0x03 Receive Stage N Address Data<br>0x04 Wait For Host Control *<br>0x05 Report Incoming Call *<br>0x06 Report Incoming Call With Address Digits *<br>0x07 Generate Inseizure Acknowledgment *<br>0x08 Send Host Acknowledgment *<br>0x09 Use Instruction List<br>0x0A Delay N Milliseconds<br><br>* The ICB has no data, so you will always use 0x00 for the value of the Data Length field. |                 |                       |
|                 | Refer to <i>Interworking TLVs (4-4)</i> if you have enabled interworking from the <i>VoIP Protocol Configure 0x00EE</i> message.                                                                                                                                                                                                                                                                                                                                                                                                                                    |                 |                       |
| :               | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                 |                       |

# Inseize Instruction List Configure 0x0029

---

**SwitchKit Name** InseizeInstrListConfig

**Type** EXS API and SwitchKit API message

**Description** **Inseize Instruction List Configure 0x0029**

Use several of these messages to form a list of configuration instruction for inseize control. You must send a separate *Inseize Instruction List Configure* message for each instruction that you add to the list.

After configuring a channel with an inseize instruction list, the CSP executes the list upon detecting an inseizure on the channel.

The last instruction in a list should always be “Wait For Host Control” or “Wait for Host Control with Answer Supervision,” where the CSP waits for further instructions from the host to continue processing.

**NOTE:** A seize instruction *cannot* be preprogrammed.

**T-ONE Card**

When performing Immediate Start on E&M trunks on a T-ONE card, the first inseize instruction must be Generate Inseize ACK.

**Related Messages**

- *Inseize Control* 0x002B (InseizeControl)
- *Call Control Instructions Query* 0x0087 (CallControlInstructionQuery)

**Sent by** Host

**Example Message**

The following example message is sent from the host to the CSP to configure an Inseize instruction list on spans 0, 1, 2, 3 - channels 0-23.

```
00 15 00 29 00 00 ff 01 02 0d 03 00 00 00 0d 03 00 03 17
ff 00 00 00
```

Clear Instruction List

```
00 15 00 29 00 00 ff 01 02 0d 03 00 00 00 0d 03 00 03 17
01 07 00 00
```

Generate Inseize Acknowledgment

```
00 15 00 29 00 00 ff 01 02 0d 03 00 00 00 0d 03 00 03 17
02 03 01 00
```

Receive Stage 1 Digits

```
00 15 00 29 00 00 ff 01 02 0d 03 00 00 00 0d 03 00 03 17
03 02 04 00
```

Generate Call Processing Event (Wink 2)

```
00 15 00 29 00 00 ff 01 02 0d 03 00 00 00 0d 03 00 03 17
04 06 00 00
```

Report Incoming Call With Address Data

```
00 15 00 29 00 00 ff 01 02 0d 03 00 00 00 0d 03 00 03 17
05 04 00 00
```

Wait For Host Control

## SwitchKit Code Configuration

```
InseizeInstrListConfig (
 Node = integer,
 Range = StartSpan:StartChan - EndSpan:EndChan,
 InstrNum = integer,
 InstrType = integer,
 InstrData1 = integer,
 InstrData2 = integer);
```

## C Structure

```
typedef struct {
 unsigned short StartSpan;
 UBYTE StartChannel;
 unsigned short EndSpan;
 UBYTE EndChannel;
 UBYTE InstrNum;
 UBYTE InstrType;
 UBYTE InstrData1;
 UBYTE InstrData2;
} XL_InseizeInstrListConfig;
```

## C++ Class

```
class XLC_InseizeInstrListConfig : public
 XLC_ChanRangeMessage {
public:
 unsigned short getStartSpan() const;
 void setStartSpan(unsigned short x);
 UBYTE getStartChannel() const;
 void setStartChannel(UBYTE x);
 unsigned short getEndSpan() const;
 void setEndSpan(unsigned short x);
 UBYTE getEndChannel() const;
 void setEndChannel(UBYTE x);
 UBYTE getInstrNum() const;
 void setInstrNum(UBYTE x);
 UBYTE getInstrType() const;
 void setInstrType(UBYTE x);
 UBYTE getInstrData1() const;
 void setInstrData1(UBYTE x);
 UBYTE getInstrData2() const;
 void setInstrData2(UBYTE x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | RESPONSE (Gray) |                       |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x0029)                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 3, 4            | Message Type (0x0029) |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 7               | Logical Node ID       |
| :               | AIB<br>Address Method<br>0x01 - Range of AEs                                                                                                                                                                                                                                                                                                                                                                                                                                         | 8, 9            | Status (MSB, LSB)     |
|                 | Number of AEs to follow<br>0x02                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 10              | Checksum              |
|                 | AEs<br>0x0D Channel (Starting)<br>0x0D Channel (Ending)                                                                                                                                                                                                                                                                                                                                                                                                                              |                 |                       |
| :               | <p>Instruction Number</p> <p>You must assign each message an instruction number, starting with instruction number 1. The <b>CSP</b> executes each message in order, by instruction number until the wait for host control instruction. Valid instruction numbers are 1 - 20.</p> <p>To clear all instructions from a channel, set the <i>Instruction Number</i> to 0xFF and <i>Instruction Type</i> to 0x00 (Null).</p>                                                              |                 |                       |
| :               | <p>Instruction Type</p> <ul style="list-style-type: none"> <li>0x00 Null</li> <li>0x01 Report Call Processing Event</li> <li>0x02 Generate Call Processing Event</li> <li>0x03 Receive Stage N Address Data</li> <li>0x04 Wait for Host Control Message</li> <li>0x05 Report Incoming Call</li> <li>0x06 Report Incoming Call with Address Digits</li> <li>0x07 Generate Inseize Acknowledgment</li> <li>0x08 Send Host Acknowledgment</li> <li>0x0A Delay N Milliseconds</li> </ul> |                 |                       |
| :               | Data[0] See Instruction Type Data table below.                                                                                                                                                                                                                                                                                                                                                                                                                                       |                 |                       |
| :               | Data[1] See Instruction Type Data table below.                                                                                                                                                                                                                                                                                                                                                                                                                                       |                 |                       |
| :               | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                 |                       |

### Instruction Type Data

| Instruction Type                    | Data 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Data 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x00 Null                           | 0x00                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 0x00                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 0x01 Report Call Processing Event   | <p>Call Processing Event</p> <p>0x00 No Event<br/>                     0x01 Off-hook<br/>                     0x02 Digits<br/>                     0x03 Wink 1<br/>                     0x04 Wink 2<br/>                     0x05 Wink 3<br/>                     0x06 Wink 4<br/>                     0x07 Wink 5<br/>                     0x08 Wink 6<br/>                     0x09 Wink 7<br/>                     0x0A Wink 8<br/>                     0x0B Dial Tone</p> <p>All other events are reserved and must be 0x00.</p>                                                                                                           | <p>0x00<br/>                     0x00<br/>                     Stage Numbers *<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00</p> <p>* This field is a bit mask. You can report multiple stages at the same time. The following are valid entries for the bits of this field:</p> <p><u>Bit</u><br/>                     0 Stage 1<br/>                     1 Stage 2<br/>                     2 Stage 3<br/>                     3 Stage 4</p> <p>Bits 4-7 are reserved and must be 0x00.</p> |
| 0x02 Generate Call Processing Event | <p>0x01 ANI Request Off-hook<br/>                     0x02 Reserved<br/>                     0x03 Wink 1<br/>                     0x04 Wink 2<br/>                     0x05 Wink 3<br/>                     0x06 Wink 4<br/>                     0x07 Wink 5<br/>                     0x08 Wink 6<br/>                     0x09 Wink 7<br/>                     0x0A Wink 8<br/>                     0x0B Backward Pulsed R2 Signal<br/>                     0x0C Backward Compelled R2 Signal<br/>                     0x0D Backward R2 Signal with Completion Event<br/>                     0x0F Backward Compelled or Pulsed R2 Signal</p> | <p>0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x00<br/>                     0x01 - 0x0F<br/>                     0x01 - 0x0F<br/>                     0x01 - 0x0F<br/>                     0x01 - 0x0F</p>                                                                                                                                                                                                                                                                                 |

Inseize Instruction List Configure 0x0029

| <b>Instruction Type</b>                       | <b>Data 0</b>                                                                                      | <b>Data 1</b>     |
|-----------------------------------------------|----------------------------------------------------------------------------------------------------|-------------------|
| 0x03 Receive Stage N Address Data             | Stage Number<br>0x01 - 0x04                                                                        | 0x00              |
| 0x04 Wait for Host Control Message            | 0x00                                                                                               | 0x00              |
| 0x05 Report Incoming Call                     | 0x00                                                                                               | 0x00              |
| 0x06 Report Incoming Call with Address Digits | 0x00                                                                                               | 0x00              |
| 0x07 Generate Inseize Acknowledgment          | 0x00                                                                                               | 0x00              |
| 0x08 Send Host Acknowledgment                 | 0x00                                                                                               | 0x00              |
| 0x0A Delay N Milliseconds                     | Delay Value, MSB:<br><br>Values are in units of 10 ms.<br>Maximum value is 10 seconds<br>(0x03E8). | Delay Value, LSB: |



# InterAppMsg

---

**Type**      SwitchKit API message

**Description**      Use the *SK\_InterAppMsg* message to send arbitrary information from one application to another. An acknowledgment is sent to the application that originally sent *SK\_InterAppMsg*.

**Important!**      You must call setDataSize before calling setData. If not, data will not be copied into the data field.

C developers must also copy the data manually. Otherwise, the data field will be truncated and the size set to 0 when the message is sent.

**Sent by**      Application

**C Structure**

```
typedef struct {
 int Target;
 union {
 UBYTE UnAckedMsg;
 UBYTE Type;
 } union21 ;
 unsigned short AckTimeout;
 int Tag1;
 int Tag2;
 int Tag3;
 int Tag4;
 char AppGroupTarget[16];
 unsigned short DataSize;
 UBYTE Data[212];
} SK_InterAppMsg;
```

**C Structure Response**

```
typedef struct {
 int Status;
 UBYTE Byte1;
 unsigned short Short1;
 int Tag1;
 int Tag2;
 int Tag3;
 int Tag4;
 unsigned short DataSize;
 UBYTE Data[228];
} SK_InterAppMsgAck;
```

```

C++ Class class SKC_InterAppMsg : public SKC_ToolkitMessage {
public:
 int getTarget() const;
 void setTarget(int x);
 UBYTE getUnAckedMsg() const;
 void setUnAckedMsg(UBYTE x);
 UBYTE getType() const;
 void setType(UBYTE x);
 unsigned short getAckTimeout() const;
 void setAckTimeout(unsigned short x);
 int getTag1() const;
 void setTag1(int x);
 int getTag2() const;
 void setTag2(int x);
 int getTag3() const;
 void setTag3(int x);
 int getTag4() const;
 void setTag4(int x);
 const char *getAppGroupTarget() const;
 void setAppGroupTarget(const char *x);
 unsigned short getDataSize() const;
 void setDataSize(unsigned short x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};

```

```

C++ Class Response class SKC_InterAppMsgAck : public SKC_ToolkitAck {
public:
 int getStatus() const;
 void setStatus(int x);
 UBYTE getByte1() const;
 void setByte1(UBYTE x);
 unsigned short getShort1() const;
 void setShort1(unsigned short x);
 int getTag1() const;
 void setTag1(int x);
 int getTag2() const;
 void setTag2(int x);
 int getTag3() const;
 void setTag3(int x);
 int getTag4() const;
 void setTag4(int x);
 unsigned short getDataSize() const;
 void setDataSize(unsigned short x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};

```

**Argument Values** The following table shows the arguments that you can modify in *InterAppMsg* :

| Argument                                 | Values and Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Target<br>AppGroupTarget                 | Write the name of the application that is to receive the message into the Target field. If you do not specify Target, AppGroupTarget must contain the text name of the application group that is to receive this message. If both Target and AppGroupTarget are specified, Target is used.                                                                                                                                                                                                                                                                                                               |
| AckTimeout                               | Specifies the minimum time the LLC will wait for an acknowledgement to an InterAppMsg before it considers the InterAppMsg to have timed-out. This field only has meaning if the Type/UnAckedMsg field is set to SK_IAM_WITH_ACK. The actual time-out may be higher than the specified value because of the mechanism that LLC uses to keep track of outstanding requests. In any case, the time-out should occur within five seconds of the time-out value specified by the originator of the message. The default value for time-out, if not specified by the originator of the message, is 15 seconds. |
| Tag 1<br>Tag 2<br>Tag 3<br>Tag 4<br>Data | The Tag and Data fields are provided only for the convenience of the user. SwitchKit simply passes along these fields without any processing. You are responsible for assigning semantic meaning to these fields. The target application will receive a <i>SK_InterAppMsg</i> with all the fields filled in as they were filled in by the sending application.                                                                                                                                                                                                                                           |
| DataSize                                 | DataSize is used to specify how many bytes of data are in the Data array. The default amount of data that can be sent is 212 bytes. It can be increased up to 64KB. Refer to <i>Send Messages Larger than Default</i> .                                                                                                                                                                                                                                                                                                                                                                                  |

| Argument                 | Values and Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Type<br>or<br>UnAckedMsg | <p>Specifies the expected behavior for this message.</p> <p>If the sender of the message fails to set this field before sending the message, the default will be <code>SK_IAM_WITH_NO_ACK</code>. In cases where acknowledgement is not expected, the recipient cannot send an acknowledgement using the <code>InterAppMsgAck</code>.</p> <p>There are three values allowed for this field:<br/> <code>SK_IAM_WITH_ACK</code><br/> <code>SK_IAM_BROADCAST</code><br/> <code>SK_IAM_WITH_NO_ACK</code></p> <p><b>SK_IAM_WITH_ACK</b> - The sender of the <code>InterAppMsg</code> expects an acknowledgement from the recipient of the message. If the recipient fails to respond, LLC will generate an acknowledgement to the originator of the <code>InterAppMsg</code> with a status of <code>SK_NO_ACK_FROM_SWITCH</code>. If the recipient is not connected to the LLC, the LLC will generate an acknowledgement to the originator of the <code>InterAppMsg</code> with a status of <code>SK_BAD_APP_NAME</code>. In any case, the acknowledgement will come in the form of a message of type <code>SK_InterAppMsgAck</code>. See <i>Acknowledgment</i>.</p> <p><b>SK_IAM_BROADCAST</b> - The sender of the <code>InterAppMsg</code> would like to send this message to all members of the <code>AppGroupTarget</code> specified in the <code>InterAppMsg</code>. There is no acknowledgement expected for this message.</p> <p><b>SK_IAM_WITH_NO_ACK</b> - The sender of the <code>InterAppMsg</code> would like to send this message to its destination without expecting an acknowledgement.</p> <p>The fields, <code>Type</code> and <code>UnAckedMsg</code>, offer the same capability and therefore the field names are used interchangeably. Only one of the fields needs to be set.</p> |

### Send Messages Larger than Default

The following example shows how to send a message of 2000 bytes, if the structure `SK_InterAppMsg` is used.

```

char storage[2000];
SK_InterAppMsg *msg = (SK_InterAppMsg *)storage;
msg->Data[0] = ...
msg->Tags = ...
char buf[2000];
int sz=2000;
int stat = sk_packMessage((MsgStruct *)msg,buf,&sz);
 if (stat != OK) {
 return stat;
 }
return sk_sendMessageWithHandler(buf,sz,tag,hf);

```

If the class *SKC\_InterAppMsg* is being used, the following example will send a message of 2000 bytes.

```
SKC_InterAppMsg msg(2000);
msg.setData...
msg.setTags...
char buf[2000];
int sz=2000;
int stat = sk_packMessage(msg.getStructPtr(),buf,&sz);
 if (stat != OK) {
 return stat;
 }
return sk_sendMessageWithHandler(buf,sz,tag,hf);
```

### Acknowledgment

To acknowledge *SK\_InterAppMsg*, the application must set the sequence number of the ACK to match the sequence number of the incoming message. For example:

```
int handleInterApp(SK_Event *evt, void *tag) {
 MsgStruct *msg = evt->IncomingMsg;
 SK_MSG_SWITCH(msg) {
 CASE_InterAppMsg(iam) {
 SK_InterAppMsgAck a;
 sk_initMsg(&a,TAG_InterAppMsgAck);
 sk_setSeqNum(&a,sk_getSeqNum(iam));
 a.Status = 0x10;
 sk_sendMsgStruct((MsgStruct *)&a, NULL, NULL);
 break;
 }
 }SK_END_SWITCH;
 return OK; //indicates that message was handled
}
```

If the C++ message classes are being used, the following example will send the ACK to an *SKC\_InterAppMsg*.

```
int handleInterApp(SK_Event *evt, void *tag) {
 SKC_Message *msg = evt->IncomingCMsg;
 SKC_MSG_SWITCH(msg) {
 CASEC_InterAppMsg(iam) {
 SKC_InterAppMsg ack;
 ack.setSeqNum(iam-getSeqNum())
 ack.setStatus(0x10);
 ack.setDataSize(0);
 ack.send();
 break;
 }
 }SKC_END_SWITCH;
 return OK; //indicates that message was handled
}
```

## IP Address Configure 0x00E7

---

**SwitchKit Name** IPAddressConfig

**Type** EXS API and SwitchKit API message

**Description** **IP Address Configure 0x00E7**

Use this message to configure the subnet mask and assign either a single IP address or all IP addresses at once. You do this by matching the number of TLVs to the number of IP addresses being assigned.

You can clear IP addresses by setting all of the IP address and subnet mask data to 0xFF. If you plan to take a card with IP addresses from one chassis and insert it into another chassis, be sure to clear all IP addresses on the card first. Otherwise, there might be a conflict with the IP addresses of the cards in the new chassis.

**NFS for DSP Series 2 card**

To use TFTP or NFS, the DSP Series 2 card must be assigned an IP Address.

**Gateway Address**

In order to assign a Gateway IP Address to a VDAC mother board or any of the four modules, you must include the Gateway IP TLV in the same message as the *IP Address Configure*. You can not assign a Gateway IP Address in a message where the Gateway IP Address is the only TLV in the message.

**Sent by** Host

**SwitchKit Code** **Configuration**

```
IPAddress Config (
 Node = integer,
 Slot = integer,
 DataType = integer,
 TLVCount = integer,
 Data = byte array);
```

**C Structure**

```
typedef struct {
 UBYTE Slot;
 UBYTE DataType;
 UBYTE TLVCount;
 UBYTE Data[221];
} XL_IPAddressConfig;
```

**C Structure Response**

```
typedef struct {
 unsigned short Status;
```

```

 UBYTE TLVCount;
 UBYTE Data[250];
} XL_IPAddressConfigAck;

```

**C++ Class**

```

class XLC_IPAddressConfig : public XLC_OutboundMessage {
public:
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
 UBYTE getDataType() const;
 void setDataType(UBYTE x);
 UBYTE getTLVCount() const;
 void setTLVCount(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};

```

**C++ Class Response**

```

class XLC_IPAddressConfigAck : public
 XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 UBYTE getTLVCount() const;
 void setTLVCount(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};

```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                 | RESPONSE (Gray) |                                                                                                                                       |
|-----------------|-----------------------------------------------------------------------------------------------------------------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------|
| Byte            | Field Description                                                                                               | Byte            | Field Description                                                                                                                     |
| 0               | Frame (0xFE)                                                                                                    | 0               | Frame (0xFE)                                                                                                                          |
| 1, 2            | Length (0xNNNN)                                                                                                 | 1, 2            | Length (0x000B)                                                                                                                       |
| 3, 4            | Message Type (0x00E7)                                                                                           | 3, 4            | Message Type (0x00E7)                                                                                                                 |
| 5               | Reserved (0x00)                                                                                                 | 5               | Reserved (0x00)                                                                                                                       |
| 6               | Sequence Number                                                                                                 | 6               | Same Sequence Number                                                                                                                  |
| 7               | Logical Node ID                                                                                                 | 7               | Logical Node ID                                                                                                                       |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs<br><hr/> Number of AEs to follow<br><hr/> AE<br>0x01 Slot | 8, 9            | Status MSB, LSB<br>0x08 Invalid TLV Combiation<br>(You must include either the 0x02 Engage IP or 0x03 Reset IP TLV when configuring.) |
| :               | Data Type (0x00) TLVs                                                                                           | 10              | TLV<br>0x04 Reset Indicator                                                                                                           |

IP Address Configure 0x00E7

|   |                                                                                                                                                                                                                                                                                                                                      |   |          |
|---|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|----------|
| : | Number of TLVs                                                                                                                                                                                                                                                                                                                       | : | Checksum |
| : | TLV<br>0x01 Module, IP Address, and Subnet Mask<br>0x02 Engage IP *<br>0x03 Reset IP *<br>0x04 Reset Indicator<br>0x05 Gateway IP<br>0x09 Ethernet Link Redundancy<br>0x14 Reset VDAC Module<br>0x0A Activate Ethernet Port<br><br>* Either the 0x02 Engage IP or 0x03 Reset IP must be included when configuring the VDAC-ONE card. |   |          |
| : | Checksum                                                                                                                                                                                                                                                                                                                             |   |          |

## IP Address Query 0x00E6

---

|                       |                                                                                                                                                                                                            |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | IPAddressQuery                                                                                                                                                                                             |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                                                                                                          |
| <b>Description</b>    | <p><b>IP Address Query 0x00E6</b></p> <p>This message allows the host to query the IP address, subnet mask, and reset indicator for Common Channel Signaling cards, VDAC cards, and VDAC VoIP Modules.</p> |
| <b>Sent by</b>        | Host                                                                                                                                                                                                       |

**SwitchKit Code**    **C Structure**

```
typedef struct {
 UBYTE Slot;
} XL_IPAddressQuery;
```

**C Structure Response**

```
typedef struct {
 unsigned short Status;
 UBYTE TLVCount;
 UBYTE Data[250];
} XL_IPAddressQueryAck;
```

**C++ Class**

```
class XLC_IPAddressQuery : public XLC_OutboundMessage {
public:
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
};
```

**C++ Class Response**

```
class XLC_IPAddressQueryAck : public
 XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 UBYTE getTLVCount() const;
 void setTLVCount(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

**EXS API Hex Format**

| MESSAGE (White)           |                                                                                                                                                            | RESPONSE (Gray) |                       |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte                      | Field Description                                                                                                                                          | Byte            | Field Description     |
| 0                         | Frame (0xFE)                                                                                                                                               | 0               | Frame (0xFE)          |
| 1, 2                      | Length (0xNNNN)                                                                                                                                            | 1, 2            | Length (0x000B)       |
| 3, 4                      | Message Type (0x00E6)                                                                                                                                      | 3, 4            | Message Type (0x00E6) |
| 5                         | Reserved (0x00)                                                                                                                                            | 5               | Reserved (0x00)       |
| 6                         | Sequence Number                                                                                                                                            | 6               | Same Sequence Number  |
| 7                         | Logical Node ID                                                                                                                                            | 7               | Logical Node ID       |
| :                         | AIB<br>Address Method<br>0x00 - Individual AEs                                                                                                             | 8, 9            | Status MSB, LSB       |
|                           | Number of AEs to follow                                                                                                                                    |                 |                       |
|                           | AE<br>0x01 Slot                                                                                                                                            |                 |                       |
| :                         | Checksum                                                                                                                                                   | 10              | Number of TLVs        |
| Response continued below: |                                                                                                                                                            |                 |                       |
| :                         | TLV<br>0x01 Module, IP Address, and Subnet Mask<br>0x04 Reset Indicator<br>0x05 Gateway IP<br>0x09 Ethernet Link Redundancy<br>0x0A Activate Ethernet Port |                 |                       |
| :                         | Checksum                                                                                                                                                   |                 |                       |

# IP Signaling Series 3 Card Configure 0x0100

---

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | ServerConfig                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>Description</b>    | <p><b>IP Signaling Series 3 Card Configure 0x0100</b></p> <p>This message is used to inform the IP Signaling Series 3 card about a CSP Matrix Series 3 Card. It sets up all of the information on the IP Signaling Series 3 card side that allows for the connection to the CSP Matrix Series 3 Card.</p> <p>When an alarm.log file is created for this message, the logical node ID, 0xFE, is not shown in the log. Instead, this message shows an API logical node ID in the alarm.log.</p> |
| <b>Sent by</b>        | Host                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>SwitchKit Code</b> | The syntax for the IP Signaling Series 3 Card Configure message corresponds to the fields of the ServerConfig message.                                                                                                                                                                                                                                                                                                                                                                        |

## C Structure

```
typedef struct {
 UBYTE AddrInfo[4];
 unsigned short ExpandedNode;
 UBYTE reserved23[2];
 unsigned short ObjectType;
 unsigned short ObjectInstanceID;
 UBYTE reserved29[68];
 UBYTE TLVCount;
 UBYTE TLVData[172];
} XL_ServerConfig;
```

## C Structure Response

```
typedef struct {
 unsigned short Status;
 UBYTE reserved6[11];
 UBYTE AddrInfo[4];
 unsigned short ExpandedNode;
 UBYTE reserved23[2];
 unsigned short ObjectType;
 unsigned short ObjectInstanceID;
 UBYTE reserved29[241];
} XL_ServerConfigAck;
```

## C++ Class

```
class XLC_ServerConfig : public XLC_OutboundMessage {
public:
 const UBYTE *getAddrInfo() const;
 UBYTE *getAddrInfo();
};
```

```
void setAddrInfo(UBYTE *x);
unsigned short getExpandedNode() const;
void setExpandedNode(unsigned short x);
unsigned short getObjectType() const;
void setObjectType(unsigned short x);
unsigned short getObjectInstanceID() const;
void setObjectInstanceID(unsigned short x);
UBYTE getTLVCount() const ;
void setTLVCount(UBYTE x);
const UBYTE *getTLVData() const;
UBYTE *getTLVData();
void setTLVData(UBYTE *x);
};
```

### **C++ Class Response**

```
class XLC_serverConfigAck : public
XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 const UBYTE *getAddrInfo() const;
 UBYTE *getAddrInfo();
 void setAddrInfo(UBYTE *x);
 unsigned short getExpandedNode() const;
 void setExpandedNode(unsigned short x);
 unsigned short getObjectType() const;
 void setObjectType(unsigned short x);
 unsigned short getObjectInstanceID() const;
 void setObjectInstanceID(unsigned short x);
};
```

**EXS Hex API Format**

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                                                                                                                    | RESPONSE (Gray) |                                                                          |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------------------------------------------------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                                                                                                                                  | Byte            | Field Description                                                        |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                                                                                                                       | 0               | Frame (0xFE)                                                             |
| 1, 2            | Length (0xNNNN)                                                                                                                                                                                                                                                                                                                                                                                    | 1, 2            | Length (0xNNNN)                                                          |
| 3, 4            | Message Type (0x0100)                                                                                                                                                                                                                                                                                                                                                                              | 3, 4            | Message Type, MSB (0x0100)                                               |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                                                                                                                    | 5               | Reserved (0x00)                                                          |
| 6               | Sequence Number                                                                                                                                                                                                                                                                                                                                                                                    | 6               | Sequence Number (same as message)                                        |
| 7               | Logical Node ID (0xFE)                                                                                                                                                                                                                                                                                                                                                                             | 7               | Logical Node ID (0xFE)                                                   |
| :               | <b>AIB</b><br>Address Method<br>0x00 - Individual AEs<br>Number of AEs to follow<br>AEs<br>0x48 Ring Communication Link<br>(If you use AIB 0x52, you must also use 0x53)<br>0x53 Object Type<br>(You can use 0x53 by itself)                                                                                                                                                                       | 8, 9            | Status MSB, LSB                                                          |
| :               | Number of TLVs to Follow                                                                                                                                                                                                                                                                                                                                                                           | 10              | AIB (Individual AEs)<br>0x48 Ring Communication Link<br>0x53 Object Type |
|                 | TLVs<br>0x0001 Add IP Signaling Series 3 Card<br>0x0002 Remove IP Signaling Series 3 Card<br>0x0003 Matrix ID<br>0x0004 Remove Matrix<br>0x0005 IP Signaling Series 3 Card Slot Number<br>0x0008 IP Signaling Series 3 Card Compatibility<br>Tag<br>0x000A Add Poll Interval<br>0x000C Matrix IP Address<br>0x000E IP Signaling Series 3 Card Interface<br>Service State<br>0x000F Link Down Timer |                 |                                                                          |
| :               | Checksum                                                                                                                                                                                                                                                                                                                                                                                           | :               | Checksum                                                                 |

# IP Signaling Series 3 Card Host Poll 0x0104

---

**Type** EXS API message

**Description** **IP Signaling Series 3 Card Host Poll 0x0104**

This message is sent unsolicited from the IP Signaling Series 3 card to the host on a periodic basis to allow the host to monitor the status of the IP Signaling Series 3 card. The default poll interval is one second.

**Sent by** CSP

**EXS Hex API Format**

| MESSAGE (White) |                                                                                                                                             | RESPONSE (Gray) |                                   |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------------------|
| Byte            | Field Description                                                                                                                           | Byte            | Field Description                 |
| 0               | Frame (0xFE)                                                                                                                                | 0               | Frame (0xFE)                      |
| 1, 2            | Length (0xNNNN)                                                                                                                             | 1, 2            | Length (0x000B)                   |
| 3, 4            | Message Type (0x0104)                                                                                                                       | 3, 4            | Message Type (0x0104)             |
| 5               | Reserved (0x00)                                                                                                                             | 5               | Reserved (0x00)                   |
| 6               | Sequence Number                                                                                                                             | 6               | Sequence Number (same as message) |
| 7               | Logical Node ID                                                                                                                             | 7               | Logical Node ID                   |
| :               | <b>AIB</b><br>Address Method<br>0x00 - Individual AEs<br>Number of AEs to follow<br>AEs<br>0x48 Ring Communication Link<br>0x53 Object Type | :               | <b>AIB</b><br>Same as message     |
| :               | Number of TLVs to Follow (0x01)                                                                                                             | :               | Checksum                          |
| :               | TLV<br>0x0021 IP Signaling Series 3 Card Host Poll                                                                                          |                 |                                   |
| :               | Checksum                                                                                                                                    |                 |                                   |

# IP Signaling Series 3 Card Query 0x0101

---

**SwitchKit Name**    IPCallServerQuery

**Type**                EXS API and SwitchKit API message

**Description**        **IP Signaling Series 3 Card Query 0x0101**

This message is used by the host to query the details of the configuration data and the status of the IP Signaling Series 3 card interface.

**Message Type**        Configuration Query

**Sent By**              Host

**EXS Hex API Format**

| Message (White) |                                                                                                                                                                                               | Response (Gray) |                                   |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------------------|
| Byte            | Field Description                                                                                                                                                                             | Byte            | Field Description                 |
| 0               | Frame (0xFE)                                                                                                                                                                                  | 0               | Frame, (0xFE)                     |
| 1, 2            | Length (0x0013)                                                                                                                                                                               | 1, 2            | Length (0xNNNN)                   |
| 3, 4            | Message Type (0x0101)                                                                                                                                                                         | 3, 4            | Message Type (0x0101)             |
| 5               | Reserved (0x00)                                                                                                                                                                               | 5               | Reserved (0x00)                   |
| 6               | Sequence Number                                                                                                                                                                               | 6               | Sequence Number (same as message) |
| 7               | Logical Node ID (0xFE)                                                                                                                                                                        | 7               | Logical Node ID (0xFE)            |
|                 |                                                                                                                                                                                               | 8, 9            | Status MSB, LSB                   |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs<br><hr/> Number of AEs to follow<br><hr/> AEs<br>0x48 Ring Communication Link<br>0x53 Object Type                                       | :               | <u>AIB</u><br>Same as message     |
| :               | Query Type<br>0x01 Query List of IP Signaling Series 3 Cards Configured<br>The query response message returns a list of configured IP Signaling Series 3 cards using the Add Call Server TLV. | :               | Number of TLVs to Follow          |

IP Signaling Series 3 Card Query 0x0101

|   |                              |   |                                                                                                                                                                                                                                                                                |
|---|------------------------------|---|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | 0x02 Query Basic Information | : | TLVs<br>0x0005 IP Signaling Series 3 Card Slot Number<br>0x0007 Reserved<br>0x0008 IP Signaling Series 3 Card Compatibility Tag<br>0x000A Add Poll Interval<br>0x000C Matrix IP Address<br>0x000E IP Signaling Series 3 Card Interface Service State<br>0x000F Link Down Timer |
| : | Reserved (0x00)              | : |                                                                                                                                                                                                                                                                                |
| : | Checksum                     | : | Checksum                                                                                                                                                                                                                                                                       |

# IP Signaling Series 3 Card Status Report 0x0105

---

**Type** EXS API message

**Description** **IP Signaling Series 3 Card Status Report 0x0105**

This message is sent unsolicited from the IP Signaling Series 3card to the host in different scenarios. This is generally caused by some Matrix interaction of which the host requires notification.

**Sent by** CSP

**EXS Hex API Format**

| MESSAGE (White) |                                                                                                                                                     | RESPONSE (Gray) |                                   |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------------------|
| Byte            | Field Description                                                                                                                                   | Byte            | Field Description                 |
| 0               | Frame (0xFE)                                                                                                                                        | 0               | Frame (0xFE)                      |
| 1, 2            | Length (0xNNNN)                                                                                                                                     | 1, 2            | Length (0xNNNN)                   |
| 3, 4            | Message Type (0x0105)                                                                                                                               | 3, 4            | Message Type (0x0105)             |
| 5               | Reserved (0x00)                                                                                                                                     | 5               | Reserved (0x00)                   |
| 6               | Sequence Number                                                                                                                                     | 6               | Sequence Number (same as message) |
| 7               | Logical Node ID                                                                                                                                     | 7               | Logical Node ID                   |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs<br><br>Number of AEs to follow<br><br>AEs<br>0x48 Ring Communication Link<br>0x53 Object Type | :               | <u>AIB</u><br>Same as message     |
| :               | Number of TLVs to Follow<br>0x01                                                                                                                    | :               | Checksum                          |
| :               | TLV<br>0x0020 IP Signaling Series 3 Card Interface Status                                                                                           |                 |                                   |
| :               | Checksum                                                                                                                                            |                 |                                   |

# IP Socket Configure 0x00F0

---

|                       |                                                                                                                                                                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | IPSocketConfig                                                                                                                                                                                                                      |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                                                                                                                                   |
| <b>Description</b>    | <p><b>IP Socket Configure 0x00F0</b></p> <p>This message is used for TCP/IP socket registration and operations. The message registers the TCP/IP socket, opens or closes a socket, and prohibits or allows traffic on a socket.</p> |
| <b>Sent by</b>        | Host                                                                                                                                                                                                                                |

## SwitchKit Code Configuration

```
IPSocketConfig (
 Node = integer,
 Slot = integer,
 DataType = integer,
 TLVCount = integer,
 Data = string);
```

## C Structure

```
typedef struct {
 BaseFields Base;
 UBYTE Slot;
 UBYTE DataType;
 UBYTE TLVCount;
 UBYTE Data[221];
} XL_IPSocketConfig;
```

## C++ Class

```
class XLC_IPSocketConfig : public XLC_OutboundMessage {
public:
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
 UBYTE getDataType() const;
 void setDataType(UBYTE x);
 UBYTE getTLVCount() const;
 void setTLVCount(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                                                                                                   | RESPONSE (Gray) |                       |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                                                                                                                                                                 | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                                                                                                                                                                      | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)                                                                                                                                                                                                                                                                                   | 1, 2            | Length (0x000B)       |
| 3, 4            | Message Type (0x00F0)                                                                                                                                                                                                                                                                             | 3, 4            | Message Type (0x00F0) |
| 5               | Reserved (0x00)                                                                                                                                                                                                                                                                                   | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                                                                                                                                                                   | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                                                                                                                                                                   | 7               | Logical Node ID       |
| :               | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs<br><hr/> Number of AEs to follow<br><hr/> AE<br>0x01 Slot                                                                                                                                                                                   | 8, 9            | Status MSB, LSB       |
| :               | Data Type (TLVs)                                                                                                                                                                                                                                                                                  | 10              | Checksum              |
| :               | TLV<br>0x01 Client Socket Register by IP<br>0x02 Server Socket Register by IP<br>0x03 Client Socket Register by Name<br>0x04 Server Socket Register by Name<br>0x05 Unregister Socket<br>0x06 Open Socket<br>0x07 Close Socket<br>0x08 Prohibit Traffic on Socket<br>0x09 Allow Traffic on Socket |                 |                       |
| :               | Checksum                                                                                                                                                                                                                                                                                          |                 |                       |

# IP Socket Query 0x00F1

---

|                       |                                                                                      |
|-----------------------|--------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | IPSocketQuery                                                                        |
| <b>Type</b>           | EXS API and SwitchKit API message                                                    |
| <b>Description</b>    | <b>IP Socket Query 0x00F1</b><br>Use this message to query IP socket configurations. |
| <b>Sent by</b>        | Host                                                                                 |
| <b>SwitchKit Code</b> | <b>C Structure</b>                                                                   |

```
typedef struct {
 UBYTE Slot;
 UBYTE DataType;
 UBYTE TLVCount;
 UBYTE Data[221];
} XL_IPSocketQuery;
```

### C Structure Response

```
typedef struct {
 unsigned short Status;
 UBYTE TLVCount;
 UBYTE Data[250];
} XL_IPSocketQueryAck;
```

### C++ Class

```
class XL_IPSocketQuery : public XLC_OutboundMessage {
public:
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
 UBYTE getDataType() const;
 void setDataType(UBYTE x);
 UBYTE getTLVCount() const;
 void setTLVCount(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

### C++ Class Response

```
class XL_IPSocketQueryAck : public
 XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 UBYTE getTLVCount() const;
 void setTLVCount(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

};

**EXS API Hex Format**

| MESSAGE (White) |                              | RESPONSE (Gray) |                       |
|-----------------|------------------------------|-----------------|-----------------------|
| Byte            | Field Description            | Byte            | Field Description     |
| 0               | Frame (0xFE)                 | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)              | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x00F1)        | 3, 4            | Message Type (0x00F1) |
| 5               | Reserved (0x00)              | 5               | Reserved (0x00)       |
| 6               | Sequence Number              | 6               | Same Sequence Number  |
| 7               | Logical Node ID              | 7               | Logical Node ID       |
| :               | AIB<br>0x00 - Individual AEs | 8, 9            | Status MSB, LSB       |
|                 | Number of AEs to follow      |                 |                       |
|                 | AE<br>0x01 Slot              |                 |                       |
| :               | Data Type                    | 10              | Checksum              |
| :               | Number of TLVs               |                 |                       |
| :               | TLV:<br>0x0B Socket ID       |                 |                       |
| :               | Checksum                     |                 |                       |

## IP Socket Status Report 0x00F2

---

|                       |                                                                                                                                                                               |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | IPSocketStatusReport                                                                                                                                                          |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                                                                             |
| <b>Description</b>    | <b>IP Socket Status Report 0x00F2</b><br>This message reports the status of a socket. The CSP sends this message to the host when a socket becomes connected or disconnected. |
| <b>Sent by</b>        | CSP                                                                                                                                                                           |

**SwitchKit Code**    **C Structure**

```
typedef struct {
 UBYTE Slot;
 UBYTE DataType;
 UBYTE TLVCount;
 UBYTE Data[221];
} XL_IPSocketStatusReport;
```

**C++ Class**

```
class XLC_IPSocketStatusReport : public
 XLC_InboundMessage {
public:
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
 UBYTE getDataType() const;
 void setDataType(UBYTE x);
 UBYTE getTLVCount() const;
 void setTLVCount(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                            | RESPONSE (Gray) |                       |
|-----------------|----------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                          | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                               | 0               | Frame (0xFE)          |
| 1, 2            | Length (0xNNNN)                                                            | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x00F2)                                                      | 3, 4            | Message Type (0x00F2) |
| 5               | Reserved (0x00)                                                            | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                            | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                            | 7               | Logical Node ID       |
| :               | AIB<br>0x00 - Individual AEs<br>Number of AEs to follow<br>AE<br>0x01 Slot | 8, 9            | Status MSB, LSB       |
| :               | Data Type (TLVs)                                                           | 10              | Checksum              |
| :               | Number of TLVs                                                             |                 |                       |
| :               | TLV:<br>0x0A Socket Status                                                 |                 |                       |
| :               | Checksum                                                                   |                 |                       |

# ISDN Interface Configure 0x0060

---

**SwitchKit Name** ISDNInterfaceConfig

**Type** EXS API and SwitchKit API message

**Description** **ISDN Interface Configure 0x0060**

Use this message to configure general options to an ISDN interface. It should be sent immediately following the *D Channel Assign* message.

**NOTE:** PCM encoding for channels controlled by the D channel is defaulted based on the network signaling variant. E-1 spans configured for Euro-ISDN default to a-law, while all other variants default to  $\mu$ -law. For more information, see the message *PCM Encoding Format Configure (0x00D9)*.

**Sent by** Host

**SwitchKit Code** **Configuration**

```
ISDNInterfaceConfig (
 Node = integer,
 Span = integer,
 Channel = integer,
 Entity = integer,
 Data = byte array);
```

## C Structure

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
 UBYTE Entity;
 UBYTE Data[222];
} XL_ISDNInterfaceConfig;
```

## C++ Class

```
class XLC_ISDNInterfaceConfig : public
 XLC_OutboundMessage {
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 UBYTE getEntity() const;
 void setEntity(UBYTE x);
 int getMsgSize() const;
 void setMsgSize(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

**Overview of message** The following table provides an overview of this message. The table following it, provides the detail for each byte.

| MESSAGE (White) |                            | RESPONSE (Gray) |                       |
|-----------------|----------------------------|-----------------|-----------------------|
| Byte            | Field Description          | Byte            | Field Description     |
| 0               | Frame (0xFE)               | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)            | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x0060)      | 3, 4            | Message Type (0x0060) |
| 5               | Reserved (0x00)            | 5               | Reserved (0x00)       |
| 6               | Sequence Number            | 6               | Same Sequence Number  |
| 7               | Logical Node ID            | 7               | Logical Node ID       |
| :               | AIB (starting with byte 0) | 8, 9            | Status (MSB, LSB)     |
| :               | :                          | 10              | Checksum              |
| :               | :                          |                 |                       |
| :               | Entity                     |                 |                       |
| :               | Data [0]                   |                 |                       |
| :               | :                          |                 |                       |
| :               | Checksum                   |                 |                       |

**EXS API Hex Format - Detailed**

| MESSAGE (White) |                                     | RESPONSE (Gray) |                                                                                    |
|-----------------|-------------------------------------|-----------------|------------------------------------------------------------------------------------|
| Byte            | Field Description                   | Byte            | Field Description                                                                  |
| 0               | Frame (0xFE)                        | 0               | Frame (0xFE)                                                                       |
| 1, 2            | Length (0xNNNN)                     | 1, 2            | Length (0x0007)                                                                    |
| 3, 4            | Message (0x0060)                    | 3, 4            | Message Type (0x0060)                                                              |
| 5               | Reserved (0x00)                     | 5               | Reserved (0x00)                                                                    |
| 6               | Sequence Number                     | 6               | Same Sequence Number                                                               |
| 7               | Logical Node ID                     | 7               | Logical Node ID                                                                    |
|                 |                                     | 8, 9            | Status (MSB, LSB)<br>0x99 - Take D channel out of service before reconfiguring it. |
| :               | <u>AIB</u><br>0x00 - Individual AEs | 10              | Checksum                                                                           |
|                 | Number of AEs to follow             |                 |                                                                                    |
|                 | AE<br>0x0D Channel                  |                 |                                                                                    |

## ISDN Interface Configure 0x0060

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | <p>Entity</p> <ul style="list-style-type: none"><li>0x01* Connection Type</li><li>0x02 Options (bit mask)</li><li>0x03 D Channel Physical Medium</li><li>0x03 Span/Channel/Subrate Length</li><li>0x04 HDLC Bit Polarity</li><li>0x05 Network Side Layer 2</li><li>0x06 B Channel Selection Mode</li><li>0x07 Location</li><li>0x08 Layer 4 <i>Channel Release Request</i> on ISDN Disconnect</li><li>0x09 Protocol Discriminator for Maintenance Messages</li><li>0x0A B Channel Encoding for Transmission</li><li>0x0B Load Information Library Entry</li><li>For ISDN only. DASS 2/DPNSS has a different format and is documented in the DASS 2/DPNSS Data ICB area of this Publication Set.</li><br/><li>0x0C Congestion Threshold</li><li>0x0D Fill Mask Functionality</li><li>0x0F Super Rate D Channel</li><li>Data[0] Reserved</li><li>Data[1] Number of D Channels (DSOs) 0x02 - 0x04</li></ul> <p>*If you use the value 0x01 (Connection Type) in the <i>Entity</i> field, you set all B channel parameters and the parameters below to their respective default values:</p> <ul style="list-style-type: none"><li>PCM Encoding Format</li><li>Release Modes</li><li>Answer Supervision Mode</li></ul> <p>If you must modify the default PPL Configuration Bytes or PPL Timers, you must first change the Connection Type.</p> |
|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | <p>Data[0]</p> <p>The meaning of the <i>Data</i> fields depends on the value of the <i>Entity</i> field. Valid entries for this field are listed below by <i>Entity</i> field value. Default values are marked with an asterisk (*).</p> <p>0x01 Connection Type<br/>Data[0] 0x00 (Reserved)</p> <p>Data[1]</p> <p><i>User-Side Variants</i></p> <p>0x01* Lucent 4ESS Q.931 PRI<br/>0x02 Lucent 5ESS Q.931 PRI (Custom)<br/>0x03 Northern Telecom DMS-100 Q.931 PRI (Custom)<br/>0x04 Northern Telecom DMS-250 Q.931 PRI (Custom)<br/>0x05 AUSTEL<br/>0x06 JATE (INS 1500)<br/>0x07 Euro-ISDN PRI (Includes French and German Delta)<br/>0x08 Reserved<br/>0x09 NI 2 User Side Connection endpoint variant<br/>0x0A User Side LAPD variant<br/>0x0B Reserved<br/>0x0C UK DPNSS<br/>0x0D* UK DASS 2 (This is the default, if the card is DASS 2/DPNSS.)</p> <p><i>Network-Side Variants</i></p> <p>0x11-0x16 Reserved<br/>0x17 Euro ISDN PRI<br/>0x19 NI 2 Network Side endpoint variant<br/>0x1A Network Side LAPD variant</p> <p>Changing the connection type modifies the PPL configuration bytes and timers to the default values for the type of CSP indicated. Therefore, change the PPL configuration bytes and timers after sending the PPL Interface Configuration message, which changes the connection type.</p> <p>0x02 Options (Bit Mask)<br/>Data[0] 0x00 (Reserved)<br/>Data[1] This field is a bit mask, where 0=No and 1=Yes:<br/>Bit 0 Send Exact I2 Frames to the Host in a <i>Diagnostic Indication</i> message<br/>Bit 1 Reserved<br/>Bit 2 DPNSS Virtual Channel DLC Initialization disable.<br/>(Disables the initiation of a SABMR on the virtual channels.)<br/>0 = Enabled (default)<br/>1 = Disabled</p> <p>0x03 D Channel Physical Medium<br/>Data[0] 0x00 (Reserved)<br/>Data[1] 0x01 64 kbps (T1/E1/J1)</p> <p>0x04 HDLC Bit Polarity<br/>Data[0] 0x00 (Reserved)<br/>Data[1] 0x00* Normal<br/>0x01 Inverted (typically used with T1 D4 and AMI line coding)</p> |
|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## ISDN Interface Configure 0x0060

|         |                                                                                                           |
|---------|-----------------------------------------------------------------------------------------------------------|
| 0x05    | Network-Side Layer 2                                                                                      |
| Data[0] | 0x00 (Reserved)                                                                                           |
| Data[1] | 0x00* User Side                                                                                           |
| 0x01    | Network Side (C/R Bit Inverted)                                                                           |
|         | If the Connection Type is set to a Network Side variant, this option is automatically set to Network Side |
| 0x06    | B Channel Selection Mode                                                                                  |
| Data[0] | 0x00 (Reserved)                                                                                           |
| Data[1] | 0x00 Disabled                                                                                             |
| 0x01*   | Linear Clockwise                                                                                          |
| 0x02    | Linear Counter Clockwise                                                                                  |
| 0x03    | Circular Clockwise                                                                                        |
| 0x04    | Circular Counter Clockwise                                                                                |
| 0x07    | Location                                                                                                  |
| Data[0] | 0x00 (Reserved)                                                                                           |
| Data[1] | 0x00* User                                                                                                |
| 0x01    | Private network serving local user                                                                        |
| 0x02    | Public network serving local user                                                                         |
| 0x03    | Transit network                                                                                           |
| 0x04    | Public network serving remote user                                                                        |
| 0x05    | Private network serving remote user                                                                       |
| 0x07    | International network                                                                                     |
| 0x0A    | Network beyond interworking point                                                                         |

# ISDN Interface Configure 0x0060

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x08 | <p>Layer 4 Channel Release Request on ISDN Disconnect</p> <p>Data[0] 0x00 (Reserved)</p> <p>Data[1] 0x00* Do not send host Channel Release Request on ISDN Disconnect<br/>         0x01 Send host Channel Release Request on ISDN Disconnect.<br/>         The host must respond to a Channel Release Request with a <i>Release With Data</i> message within 20 seconds.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0x09 | <p>Protocol Discriminator Value for Maintenance Messages</p> <p>Data[0] 0x00 (Reserved)</p> <p>Data[1] 0x03</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 0x0A | <p>B Channel Encoding for Transmission</p> <p>Data[0] 0x00 (Reserved)</p> <p>Data[1] 0x00* Channel Number<br/>         0x01 Slot Map</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 0x0B | <p>Load Information Library Entry</p> <p>Data[0] Entry Number (0x00-0x1D)</p> <p>Data[1] IE Type<br/>         0x01 Q.931 IE</p> <p>Include the following data fields only if Data[1] above is 0x01 (Q.931 IE type):</p> <p>Data[2] Total IE Length (0x00-0x1C)</p> <p>Data[3] IE Data</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 0x0C | <p>Congestion Threshold</p> <p>Data [0-3] Level 2 burst threshold.<br/>         Range 1-10000 messages (Default = 250 messages)</p> <p>Data [4-7] Level 2 average threshold<br/>         Range 1-10000 messages (Default = 1300 messages)</p> <p>Data [8-11] Abatement threshold<br/>         Range 1-10000 messages (Default = 300 messages)</p> <p>Data [12-15] Burst time window<br/>         Range 1-10000 seconds (Default = 1 second)</p> <p>Data [16-19] Number of samples in average<br/>         Range 0-19 burst time windows (Default =10 burst time windows)</p> <p>Data [20-23] Abatement window<br/>         Range 1-10000 seconds (Default = 5 seconds)</p> <p>Data [24--27] Level 1 burst threshold<br/>         Range 1-10000 messages (Default = 200 messages)</p> <p>Data [28-31] Level 1 average threshold<br/>         Range 1-10000 messages (Default = 1100 messages)</p> |
|      | <p>For the Data Field values above, each value consists of four bytes (one "long word")</p> <p>* default value</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 0x0D | <p>Fill Mask Functionality</p> <p>Data [0] Range</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| :    | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

# ISDN Query 0x0063

---

**SwitchKit Name** ISDNQuery

**Type** EXS API and SwitchKit API message

**Description** **ISDN Query 0x0063**

Use this message to query specific parameters configurable by the *ISDN Terminal Configure*, *ISDN Interface Configure*, and *PPL Configure* messages. Hosts applications should consider that future configuration options will be added to this query message, thus changing the length.

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 UBYTE AddrInfo[30];
 UBYTE Type;
 UBYTE SubType;
} XL_ISDNQuery;
```

**C Structure Response**

```
typedef struct {
 unsigned short Status;
 UBYTE Data[251];
} XL_ISDNQueryAck;
```

**C++ Class**

```
class XLC_ISDNQuery : public XLC_OutboundMessage {
public:
 const UBYTE *getAddrInfo() const;
 UBYTE *getAddrInfo();
 void setAddrInfo(UBYTE *x);
 XBYTE getSpan() const;
 void setSpan(XBYTE x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 UBYTE getSlot() const;
 void setSlot(UBYTE x);
 UBYTE getType() const;
 void setType(UBYTE x);
 UBYTE getSubType() const;
 void setSubType(UBYTE x);
};
```

**C++ Class Response**

```
class XLC_ISDNQueryAck : public XLC_AcknowledgeMessage {
public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
```

```

const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};

```

**EXS API Hex Format**

| MESSAGE |                                 | RESPONSE |                           |
|---------|---------------------------------|----------|---------------------------|
| Byte    | Field Description               | Byte     | Field Description         |
| 0       | Frame (0xFE)                    | 0        | Frame (0xFE)              |
| 1, 2    | Length (0x00NN)                 | 1, 2     | Length (0x0007)           |
| 3, 4    | Message Type (0x0063)           | 3, 4     | Message Type (0x0063)     |
| 5       | Reserved (0x00)                 | 5        | Reserved (0x00)           |
| 6       | Sequence Number                 | 6        | Same Sequence Number      |
| 7       | Logical Node ID                 | 7        | Logical Node ID           |
| :       | AIB                             | 8, 9     | Status (MSB, LSB)         |
|         | 0x00 - Individual AEs           | 10       | Data[0] (See table below) |
|         | Number of AEs to follow         | :        | Checksum                  |
|         | AE<br>0x01 Slot<br>0x0D Channel |          |                           |

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| : | <p>Query Type</p> <ul style="list-style-type: none"> <li>0x01 General Interface Options<br/>See <i>ISDN Interface Configure</i> message (0x60)</li> <li>0x02 Terminal Configuration Data</li> <li>0x03 Assigned Protocols</li> <li>0x04 IE Library</li> <li>0x05 HDLC Statistics</li> <li>0x06 User Service Order Profile (USOP) Database</li> <li>0x08 Congestion Data</li> <li>0x0E Configured D Channels</li> <li>0x0D Super Rate LAPD</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| : | <p>Query Subtype</p> <p>The meaning of the <i>Query Subtype</i> field depends on the value of the <i>Query Type</i> field. Valid entries for this field are listed below by <i>Query Type</i> field value.</p> <ul style="list-style-type: none"> <li>0x01 General Interface Options <ul style="list-style-type: none"> <li>0x00 None</li> </ul> </li> <li>0x02 Terminal Configuration Data <ul style="list-style-type: none"> <li>0x00 LAPD Info</li> </ul> </li> <li>0x03 Assigned Protocols <ul style="list-style-type: none"> <li>0x00 None</li> </ul> </li> <li>0x04 IE Library <ul style="list-style-type: none"> <li>Entry Number (User-defined)</li> </ul> </li> <li>0x05 HDLC Statistics <ul style="list-style-type: none"> <li>0x00 None</li> </ul> </li> <li>0x08 Congestion Data <ul style="list-style-type: none"> <li>0x00 None</li> </ul> </li> <li>0x0E Configured D Channels <ul style="list-style-type: none"> <li>0x00 None</li> </ul> </li> </ul> |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10 | <p>Data[0]</p> <p><b>0x01 General Interface Options</b></p> <p>Data[0, 1] Connection Type<br/>Data[0] 0x00 (Reserved)</p> <p>Data[1]</p> <p><i>User-Side Variants</i></p> <p>0x01* Lucent 4ESS Q.931 PRI<br/>0x02 Lucent 5ESS Q.931 PRI (Custom)<br/>0x03 Northern Telecom DMS-100 Q.931 PRI (Custom)<br/>0x04 Northern Telecom DMS-250 Q.931 PRI (Custom)<br/>0x05 AUSTEL<br/>0x06 JATE (INS 1500)<br/>0x07 Euro-ISDN PRI (Includes French and German Delta)<br/>0x08 Reserved<br/>0x09 NI 2 User Side Connection endpoint variant<br/>0x0A User Side LAPD variant<br/>0x0B Reserved<br/>0x0C UK DPNSS<br/>0x0D* UK DASS 2 (This is the default, if the card is DASS 2/DPNSS.)</p> <p><i>Network-Side Variants</i></p> <p>0x11-0x16Reserved<br/>0x17 Euro ISDN PRI<br/>0x19 NI 2 Network Side endpoint variant<br/>0x1A Network Side LAPD variant</p> <p>Data[2, 3] Options (bit mask)<br/>Data[2] Reserved (0x00)<br/>Data[3] This field is a bit mask, where 0=No and 1=Yes:<br/>Bit 0Send Exactly12 Frames to the Host<br/>in a <i>Diagnostic Indication</i> message.<br/>Bit 1DASS2 Ignore Receive Sequence Number<br/>Bits 2-7Reserved (must be 0)</p> <p>Data[4, 5] D Channel Physical Medium<br/>Data[4] 0x00 (Reserved)<br/>Data[5] 0x01 64 kbps (T1/E1/J1)</p> <p>Data[6, 7] HDLC Bit Polarity<br/>Data[6] 0x00 (Reserved)<br/>Data[7] 0x00*Normal<br/>0x01Inverted (typically used with T1 D4 and AMI line coding)</p> <p>Data[8, 9] Network Side Layer 2<br/>Data[8] 0x00 (Reserved)<br/>Data[9] 0x00*User Side<br/>0x01 Network Side (C/R Bit Inverted)</p> |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Data[10, 11] B Channel Selection Mode  
 Data[10] 0x00 (Reserved)  
 Data[11] 0x00 Disabled  
 0x01\*Linear Clockwise  
 0x02Linear Counter Clockwise  
 0x03Circular Clockwise  
 0x04Circular Counter Clockwise

Data[12, 13] Location  
 Data[12] 0x00 (Reserved)  
 Data[13] 0x00\*User  
 0x01Private network serving local user  
 0x02Public network serving local user  
 0x03Transit network  
 0x04Public network serving remote user  
 0x05Private network serving remote user  
 0x07International network  
 0x0ANetwork beyond interworking point

Data[14, 15] Channel Release Request on ISDN Disconnect  
 Data[14] 0x00 (Reserved)  
 Data[15] 0x00\*Do not send Host Channel Release Request  
 on ISDNDisconnect  
 0x01Send Host Channel Release Request  
 on ISDN Disconnect

NOTE: The host must respond to a *Channel Release Request* with a  
*Release With Data* message within 20 seconds.

Data[16, 17] Protocol Discriminator Value for Maintenance Messages  
 Data[0] 0x00 (Reserved)  
 Data[1] 0x03

Data[18, 19] B Channel Encoding for Transmission  
 Data[0] 0x00 (Reserved)  
 Data[1] 0x00\*Channel Number  
 0x01Slot Map

**0x02 Terminal Configuration Data**

LAPD Info

Data[0, 1] Q.921 T200  
 Data[2, 3] Q.921 T201  
 Data[4, 5] Q.921 T202  
 Data[6, 7] Q.921 T203  
 Data[8, 9] Q.921 N200  
 Data[10, 11] Q.921 N201  
 Data[12, 13] Q.921 N202  
 Data[14, 15] Acknowledge Pending Timeout  
 Data[16, 17] Window Size  
 Data[18, 19] Options (bit mask)

**0x03 Assigned Protocols**

Data[0, 1] Protocol ID for component 5  
 Data[2, 3] Protocol ID for component 6  
 Data[4, 5] Protocol ID for component 7  
 Data[6, 7] Protocol ID for component 8  
 Data[8, 9] Protocol ID for component 9  
 Data[10, 11] Protocol ID for component 10  
 Data[12, 13] Protocol ID for component 11  
 Data[14, 15] Protocol ID for component 12  
 Data[16, 17] Protocol ID for component 13  
 Data[18, 19] Protocol ID for component 14

**0x04 IE Library**

Data[0] Entry Number (0–29)  
 Data[1] IE Type  
     0x01 Q.931 IEs  
 Include the following data fields only if Data[1] above is 0x01 (Q.931 IE type):  
 Data[2] Total IE Length (1–28)  
 Data[3. . .] IE Data

**0x05 HDLC Statistics**

Data[0, 1] Long Frames Detected  
 Data[2, 3] Aborts Received  
 Data[4, 5] CRC Errors  
 Data[6, 7] Short Frames  
 (Counters are reset after Query)

**0x06 USOP Database**

Eight blocks of information are reported for each USOP block configured on the D channel.  
 The format of each block is as follows:

Data[0-11] USPID  
 Data[12-21] Directory Number  
 Data[22] GSP Index  
 Data[23] Terminal Initialization Method  
     0x00 Auto FIT  
     0x01 Fixed NIT  
     0x02 Fixed FIT  
     0x80 Auto NIT with DN  
     0x81 Auto NIT with no Calling Party Number indicated  
     (use default USOP)  
 Data[24] Fixed Terminal Endpoint Identifier (TEI) (0–63)

**0x08 Congestion Data**

Message Responses:

- Data [0- 3] Level 2 burst threshold
- Data [4-7] Level 2 average threshold
- Data [8-11] Abatement threshold
- Data [12-15] Burst time window
- Data [16-19] Number of samples in average
- Data [20-23] Abatement window
- Data [24-27] Level 1 burst threshold
- Data [28-31] Level 1 average threshold

**0x0E Configured D Channels**

Message Responses:

- Data [0, 1] Maximum Number of D-Channels Configurable (MSB, LSB)
- Data [2, 3] D-Channel 1 - Logical Span ID (MSB, LSB)
- Data [4] D-Channel 1 - Channel
- Data [5] Primary Flag (0x00 if Primary, 0x01 if Secondary)
- :
- :
- Data [n -]

NOTE: D-channels that are not configured will return a value of 0xFF FF FF FF.

**0x0D Super Rate LAPD**

- Data[0,1] D Channel offset (as specified in D Channel Assign 0xC4)
- Data[2,3] Super Rate base channel (DSO)
- Data[4,5] Number of DSOs for Super Rate

## ISDN Terminal Configure 0x0062

---

|                       |                                                                                                                                                                                      |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | ISDNTerminalConfig                                                                                                                                                                   |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                                                                                    |
| <b>Description</b>    | <b>ISDN Terminal Configure 0x0062</b><br>For ISDN Series 3, this message is used to configure LAPD timers for Service Access Point ID (SAPI = 0) and Terminal Endpoint ID (TEI = 0). |
| <b>Sent by</b>        | Host                                                                                                                                                                                 |

**SwitchKit Code**    **Configuration**

```
ISDNTerminalConfig (
 Node = integer,
 Span = integer,
 Channel = integer,
 CfgType = integer,
 Data = byte array);
```

**C Structure**

```
typedef struct {
 unsigned short Span;
 UBYTE Channel;
 UBYTE CfgType;
 UBYTE Data[222];
} XL_ISDNTerminalConfig;
```

**C++ Class**

```
class XLC_ISDNTerminalConfig : public XLC_OutboundMessage
{
public:
 unsigned short getSpan() const;
 void setSpan(unsigned short x);
 UBYTE getChannel() const;
 void setChannel(UBYTE x);
 UBYTE getCfgType() const;
 void setCfgType(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                      | RESPONSE (Gray) |                       |
|-----------------|--------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                    | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                         | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)                                                                      | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x0062)                                                                | 3, 4            | Message Type (0x0062) |
| 5               | Reserved (0x00)                                                                      | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                      | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                      | 7               | Logical Node ID       |
| :               | <u>AIB</u><br>0x00 - Individual AEs<br>Number of AEs to follow<br>AE<br>0x0D Channel | 8, 9            | Status (MSB, LSB)     |

ISDN Terminal Configure 0x0062

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |    |          |
|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----------|
| : | Configuration Type<br>0x01 LAPD                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 10 | Checksum |
| : | Data LAPD (0x01)<br>Data[0] Entity<br>0x01 Q.921 T200 Timer<br>0x02 Q.921 T201 Timer<br>0x03 Q.921 T202 Timer<br>0x04 Q.921 T203 Timer<br>0x05 Q.921 N200 Timer<br>0x06 Q.921 N201 Timer<br>0x07 Q.921 N202 Timer<br>0x08 Acknowledge Pending Timeout Timer<br>0x09 Window Size (maximum outstanding messages without Acknowledgment)<br>0x0A Options<br><br>If the value of Data[0] is 0x01–0x04, 0x08:<br>Data[1] Timer Value, MSB<br>Data[2] Timer Value, LSB<br><br>If the value of Data[0] is 0x05-0x07<br>Data[1] Parameter Value<br><br>If the value of Data[0] is 0x09<br>Data[1] Window Size Value, MSB<br>Data[2] Window Size Value, LSB<br><br>If the value of Data[0] is 0x0A:<br>Data[1] Reserved (0x00)<br>Data[2] This field is a bit mask, where 0=No and 1=Yes:<br>Bit 0 Reserved<br>Bit 1 Stop SABME transmission after N200 Counter is reached<br>Bit 2 OPEN LAPD (required software license)<br>Bit 3 Enable sending of FRMR Frames<br>Bits 4-7 Reserved (must be 0) |    |          |
| : | Checksum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |    |          |

# J1 Span Configure 0x0019

---

|                       |                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SwitchKit Name</b> | J1SpanConfig                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Type</b>           | EXS API and SwitchKit API message                                                                                                                                                                                                                                                                                                                                               |
| <b>Description</b>    | <p><b>J1 Span Configure 0x0019</b></p> <p>This message is used by the host to configure the transmitted pattern of certain bits in different frames of a multi-frame over the J-ONE interface.</p> <p><b>NOTE:</b> Before changing span configuration, Dialogic ALWAYS RECOMMENDS that you de-assign the spans, assign them again and then send the new span configuration.</p> |

**Sent by** Host

**SwitchKit Code** **Configuration**

```
J1SpanConfig (
 Node = integer,
 StartSpanRange = integer,
 EndSpanRange = integer,
 ChanData = byte array,
 FrameData = byte array);
```

### C Structure

```
typedef struct {
 unsigned short StartSpanRange;
 unsigned short EndSpanRange;
 UBYTE ChanData[6];
 UBYTE FrameData[217];
} XL_J1SpanConfig;
```

### C++ Class

```
class XLC_J1SpanConfig : public XLC_SpanRangeMessage {
public:
 XBYTE getStartSpan() const;
 XBYTE getEndSpan() const;
 void setStartSpan(XBYTE s);
 void setEndSpan(XBYTE s);
 unsigned short getStartSpanRange() const;
 void setStartSpanRange(unsigned short x);
 unsigned short getEndSpanRange() const;
 void setEndSpanRange(unsigned short x);
 const UBYTE *getChanData() const;
 UBYTE *getChanData();
 void setChanData(UBYTE *x);
 const UBYTE *getFrameData() const ;
```

```
UBYTE *getFrameData();
void setFrameData(UBYTE *x);
};
```

**EXS API Hex Format**

| MESSAGE (White) |                                                                                                                                                                                                                                           | RESPONSE (Gray) |                       |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------|
| Byte            | Field Description                                                                                                                                                                                                                         | Byte            | Field Description     |
| 0               | Frame (0xFE)                                                                                                                                                                                                                              | 0               | Frame (0xFE)          |
| 1, 2            | Length (0x00NN)                                                                                                                                                                                                                           | 1, 2            | Length (0x0007)       |
| 3, 4            | Message Type (0x0019)                                                                                                                                                                                                                     | 3, 4            | Message Type (0x0019) |
| 5               | Reserved (0x00)                                                                                                                                                                                                                           | 5               | Reserved (0x00)       |
| 6               | Sequence Number                                                                                                                                                                                                                           | 6               | Same Sequence Number  |
| 7               | Logical Node ID                                                                                                                                                                                                                           | 7               | Logical Node ID       |
| :               | AIB                                                                                                                                                                                                                                       | 8, 9            | Status (MSB, LSB)     |
|                 | Address Method<br>0x01 - Range of AEs                                                                                                                                                                                                     | 10              | AIB                   |
|                 | Number of AEs to follow<br>0x02                                                                                                                                                                                                           | :               | Checksum              |
|                 | AEs<br>0x0C Logical Span (Starting)<br>0x0C Logical Span (Ending)                                                                                                                                                                         |                 |                       |
| :               | Channel Data[0]<br>Bit mask of values for channels 1–5                                                                                                                                                                                    |                 |                       |
| :               | Channel Data[1]<br>Bit mask of values for channels 6–10                                                                                                                                                                                   |                 |                       |
| :               | Channel Data[2]<br>Bit mask of values for channels 11–15                                                                                                                                                                                  |                 |                       |
| :               | Channel Data[3]<br>Bit mask of values for channels 17–21                                                                                                                                                                                  |                 |                       |
| :               | Channel Data[4]<br>Bit mask of values for channels 22–26                                                                                                                                                                                  |                 |                       |
| :               | Channel Data[5]<br>Bit mask of values for channels 27–31                                                                                                                                                                                  |                 |                       |
| :               | Frame Data[0]<br>Bit mask of values for HG1–HG5 of Frame 0                                                                                                                                                                                |                 |                       |
| :               | Frame Data[1]<br>Bit mask of values for HG1–HG5 of Frame 7                                                                                                                                                                                |                 |                       |
| :               | Frame Data[2]<br>Bit mask of values for HG1–HG5 of Frame 8                                                                                                                                                                                |                 |                       |
| :               | Frame Data[3]<br>Channel or Frame data bit mask of values for HG1–HG5 of Frame 15<br>Bit 7 Does not matter (MSB)<br>Bit 6 Does not matter<br>Bit 5 Does not matter<br>Bit 4 HG1<br>Bit 3 HG2<br>Bit 2 HG3<br>Bit 1 HG4<br>Bit 0 HG5 (LSB) |                 |                       |
| :               | Checksum                                                                                                                                                                                                                                  |                 |                       |

## Line Card Switchover 0x0024

---

**SwitchKit Name** LineCardSwitchover

**Type** EXS API and SwitchKit API message

**Description** **Line Card Switchover 0x0024**

This message initiates a switchover from an active line card or EXNET card to a standby card in redundant system. The standby cards must be configured using the *Standby Line Card Configure* message before this message can be used to initiate a switchover.

A switchover can be initiated using this message for the following cards:

- EXNET-ONE
- T-ONE
- E-ONE
- J-ONE

**Sent by** Host

**SwitchKit Code** **C Structure**

```
typedef struct {
 UBYTE OrigSlot;
 UBYTE DestSlot;
} XL_LineCardSwitchover;
```

**C++ Class**

```
class XLC_LineCardSwitchover : public XLC_OutboundMessage
{
public:
 UBYTE getOrigSlot() const;
 void setOrigSlot(UBYTE x);
 UBYTE getDestSlot() const;
 void setDestSlot(UBYTE x);
};
```

**EXS API Hex Format**

| MESSAGE |                                                                                                                                               | RESPONSE |                       |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------|----------|-----------------------|
| Byte    | Field Description                                                                                                                             | Byte     | Field Description     |
| 0       | Frame (0xFE)                                                                                                                                  | 0        | Frame (0xFE)          |
| 1, 2    | Length (0x00NN)                                                                                                                               | 1, 2     | Length (0x0007)       |
| 3, 4    | Message Type (0x0024)                                                                                                                         | 3, 4     | Message Type (0x0024) |
| 5       | Reserved (0x00)                                                                                                                               | 5        | Reserved (0x00)       |
| 6       | Sequence Number                                                                                                                               | 6        | Same Sequence Number  |
| 7       | Logical Node ID                                                                                                                               | 7        | Logical Node ID       |
| :       | <u>AIB</u><br>Address Method<br>0x00 - Individual AEs<br>Number of AEs to follow<br>0x02<br>AEs<br>0x01 Slot (Starting)<br>0x01 Slot (Ending) | 8, 9     | Status (MSB, LSB)     |
| :       | Checksum                                                                                                                                      | 10       | Checksum              |

# LLCControl

---

|                    |                                                                                                                                                                             |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Type</b>        | SwitchKit API message                                                                                                                                                       |
| <b>Purpose</b>     | Use LLCControl to manage the behavior of LLC. LLCControl can be used to dynamically modify environment variables and to manage SS7 redundancy.                              |
| <b>Description</b> | The message is used for controlling or modifying the behavior of the LLC. The message consists of a sequence of TLVs, a command TLV followed by one or more parameter TLVs. |

## Establish Redundancy

The *SK\_LLC\_CONTROL\_ESTABLISH\_REDUNDANCY\_TLV* is used to initiate synchronization between components that are configured as redundant pairs. Synchronization can only be initiated if the components have been previously configured as redundant pairs and each component is in a state where synchronization can commence. A successful reply to this message indicates that the synchronization process has been initiated. It does not indicate the success or failure of that synchronization process.

The use of this TLV requires that the ComponentType and PrimaryLNI TLV's also be included and properly defined. The only component type supported is SS7 (0x1000).

## Control Switchover

*SK\_LLC\_CONTROL\_SWITCHOVER\_COMP\_TLV* (0x0021) is used to initiate a switchover of fully synchronized, redundant components. Switchover can only be initiated if the components have been previously configured as redundant pairs and each component is in a state where switchover can occur. A successful reply to this message indicates that the switchover process has been initiated. It does not indicate the success or failure of that switchover process.

The use of this TLV requires that the Component Type and Primary LNI TLV's also be included and properly defined. The only component type supported is SS7(0x1000).

**Sent by** Application

**How to use** To use *LLCControl*, construct the message and set the TLV information.

```
SKC_LLCCControl lq;
lq.setDataSize(DataSize);
lq.setTLVCount(TLVCount);
lq.setData(data);
```

**Arguments** The following table shows the arguments you can change:

| Argument | Description                  |
|----------|------------------------------|
| DataSize | Number of bytes of the data. |
| TLVCount | Number of TLVs in the data.  |
| data     | Data bytes.                  |

**Status Argument**

The following are values that can be returned in the Status argument of *LLCControlAck*

| Value                            | Description                                                                |
|----------------------------------|----------------------------------------------------------------------------|
| SK_BAD_PARAM                     | Returned if there is an unknown command TLV.                               |
| SK_BAD_OR_MISSING_PARAM          | A required parameter TLV is missing from the LLCControl.                   |
| SK_NODE_NOT_PREVIOUSLY_DEFINED   | A node specified in a parameter TLV is unknown to the LLC.                 |
| SK_UNABLE_TO_INITIATE_SYNC       | An attempt to initiate synchronization of SS7 redundant components failed. |
| SK_UNABLE_TO_INITIATE_SWITCHOVER | An attempt to initiate switchover of SS7 redundant components failed.      |
| SK_UNKNOWN_REDUNDANT_COMPONENT   | The specified redundant component is unknown.                              |

**Command TLVs SK\_LLC\_CONTROL\_ESTABLISH\_REDUNDANCY (0x0015)**

| Byte   | Description                                               |
|--------|-----------------------------------------------------------|
| Tag    | Data[0, 1] = SK_LLC_CONTROL_ESTABLISH_REDUNDANCY (0x1015) |
| Length | Data[2,3] = 0x0000                                        |

**SK\_LLC\_CONTROL\_SWITCHOVER\_COMP\_TLV (0x0021)**

| Byte   | Description                                              |
|--------|----------------------------------------------------------|
| Tag    | Data[0, 1] = SK_LLC_CONTROL_SWITCHOVER_COMP_TLV (0x0021) |
| Length | Data[2, 3] = 0x0000                                      |

**Parameter TLVs SK\_LLC\_CONTROL\_GO\_STANDBY\_TLV (0x0016)**

This SK\_LLC\_CONTROL\_GO\_STANDBY\_TLV will cause an LLC switchover. The current Active LLC will become Standby and will inform the current Standby (if any) that it should become Active. Note that switchover behavior is dependant on the current SK\_LLC\_SWITCHBACK\_MODE environment variable. See the environment variables section for details.

| Byte   | Description                                         |
|--------|-----------------------------------------------------|
| Tag    | Data[0, 1] = SK_LLC_CONTROL_GO_STANDBY_TLV (0x0016) |
| Length | Data[2, 3] = 0x0000                                 |
| Value  | Not Applicable                                      |

**SK\_COMPONENT\_TYPE\_TLV (0x1008)**

| Byte   | Description                                 |
|--------|---------------------------------------------|
| Tag    | Data[0, 1] = SK_COMPONENT_TYPE_TLV (0x1008) |
| Length | Data[2, 3] = 0x0002                         |

| Byte  | Description                                                       |
|-------|-------------------------------------------------------------------|
| Value | Data[4, 5] = Component ID (currently only support 0x1000 for SS7) |

**SK\_PRIMARY\_LNI\_TLV (0x1006)**

| Byte   | Description                                                       |
|--------|-------------------------------------------------------------------|
| Tag    | Data[0, 1] = SK_PRIMARY_LNI_TLV (0x1006)                          |
| Length | Data[2, 3] = 0x0002                                               |
| Value  | Data[4, 5] = Component ID (currently only support 0x1000 for SS7) |

**C Structure**

```
typedef struct {
 unsigned short DataSize;
 UBYTE TLVCount;
 UBYTE Data[250];
} SK_LLCControl;
```

**C Structure Response**

```
typedef struct {
 int Status;
 UBYTE reserved21[10]
} SK_LLCControlAck;
```

**C++ Class**

```
class SKC_LLCControl : public SKC_ToolkitMessage {
public:
 unsigned short getDataSize() const;
 void setDataSize(unsigned short x);
 UBYTE getTLVCount() const;
 void setTLVCount(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getTLVData();
 void setTLVData(UBYTE *x);
};
```

**C++ Class Response**

```
class SKC_LLCControlAck : public SKC_ToolkitAck {
public:
 int getStatus() const;
 void setStatus(int x);
};
```

# LLCQuery

---

- Type** SwitchKit API message
- Purpose** Use the *LLCQuery* message to query the LLC for information. Currently, only information about configured nodes is available.
- Description** The message is for querying the LLC for node related information. Supported queries include the following:
- Logical Node IDs
  - IP address of the active CSP Matrix Series 3 Card
  - IP address of the standby CSP Matrix Series 3 Card
- Sent by** Application
- How to use** To use *LLCQuery*, construct the message and set the TLV information.
- ```
SKC_LLCQuery1q;
1q.setDataSize(DataSize);
1q.setTLVCount(TLVCount);
1q.setData(data);
```
- Arguments** The following table shows the arguments you can change in LLCQuery:

Argument	Description
DataSize	Number of bytes of the data.
TLVCount	Number of TLVs in the data.
data	Data bytes.

Status

These values can be returned in the Status argument of *SK_LLCQueryAck*.

- SK_BAD_PARAM if unknown, query TLV.
- SK_INVALID_NODE if unknown node ID is given.

TLVs You can use several TLVs with *LLCQuery*. See TLV Syntax:

Queries

- PopulationQueryTLV (0x0002)

- NodeInfoQueryTLV (0x0003)
- LLC_REDUNDANCY_QUERY_TLV (0x0005)

Responses

- NodeInfoTLV (0x8010)
- RequestedMatrixInfoTLV (0x8011)
- RequestedDeviceServerTLV (0x8012)
- RequestedKnownNodeInfoTLV (0x8013)
- *GenericLLCReport* messages

PopulationQuery TLV

The PopulationQuery TLV is used to query the LLC about any nodes it has detected. This TLV can request information on controlled nodes, known nodes, or both controlled and known nodes. A controlled node is any node directly under this LLC's control, while a known node is under another LLC's control.

The valid values for this option are:

ControlledNode (0x01)

KnownNode (0x02)

ControlledAndKnownNode (0x03)

The LLC will respond with an LLCQueryAck with zero or more NodeInfo TLVs; one for each node that matches the criteria: known or controlled. These TLVs reveal the Logical Node ID, the Node Type, and whether the link is up, down, or unknown. If the node is not controlled by this LLC, then the link status will not be known.

If no nodes match the criteria, then an LLCQueryAck will be returned with as a positive ACK with no TLVs.

NodeInfoQuery TLV

The NodeInfoQuery TLV is used to query the LLC about a specific node ID. It has one TLV field: Logical Node ID.

The LLC responds to NodeInfoQuery TLVs with an LLCQueryAck with various TLVs depending on what type of device the node is. These TLVs would be one of the following:

- RequestedMatrixInfoTLV
- RequestedDeviceServer LV

- RequestedKnownNodeInfoTLV

The above TLVs contain all the information the LLC knows about the node. For example, if the device is a standard CSP with two Matrix cards, then the ACK will contain two RequestedMatrixInfo TLVs; one for each card. However, if the node type is a device server, for example, the IP Signaling Series 3 card, the ACK will contain one RequestedDeviceServerTLV. If the node is merely a “known node”, then the RequestedKnownNodeInfoTLV will be returned regardless of what type of node it is.

SK_LLC_REDUNDANCY_QUERY_TLV (0x0005)

The SK_LLC_REDUNDANCY_QUERY_TLV is used to query the redundancy status of the primary and redundant LLCs. The LLC will respond with a *GenericLLCReport* message of type SK_LLC_REDUNDANCY for each LLC. See the *GenericLLCReport* section for details on message behavior and format.

Description	Byte (Value)
Tag	Data[0, 1] = 0x0005 (SK_LLC_REDUNDANCY_QUERY_TLV)
Length	Data[2, 3] = 0x0000
Value	There are no values for this option.

C Structure

```
typedef struct {
    unsigned short DataSize;
    UBYTE TLVCount;
    UBYTE Data[250];
} SK_LLCQuery;
```

C Structure Response

```
typedef struct {
    unsigned short DataSize;
    int Status;
    UBYTE TLVCount;
    UBYTE Data[246];
} SK_LLCQueryAck;
```

C++ Class

```
class SKC_LLCQuery : public SKC_ToolkitMessage {
public:
    unsigned short getDataSize() const;
    void setDataSize(unsigned short x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
```

LLCQuery

```
const UBYTE *getData() const;  
UBYTE *getData();  
void setData(UBYTE *x);  
};
```

C++ Class Response

```
class SKC_LLCQueryAck : public SKC_ToolkitAck {
public:
    unsigned short getDataSize() const;
    void setDataSize(unsigned short x);
    int getStatus() const;
    void setStatus(int x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

TLV Syntax

Below is the definition of all TLVs used in the LLC Query message and the LLC Query Acknowledgement (ACK) message.

PopulationQueryTypeTLV (0x0002)

Description	Byte (Value)
Tag	Data[0, 1] = 0x0002 (PopulationQueryTypeTLV)
Length	Data[2, 3] = 0x0001 (LLCPopulationQuery)
Value	Data[4] = Known Controlled/Both

NodeInfoQueryTLV (0x0003)

Description	Byte (Value)
Tag	Data[0, 1] = 0x0003 (NODEINFOQueryTLV)
Length	Data[2, 3] = 0x0002
Value	Data[4, 5] = Logical Node ID

NodeInfoTLV (0x8010)

Description	Byte (Value)
Tag	Data[0, 1] = 0x8001 (NodeInfoTLV)
Length	Data[2, 3] = 0x0003

Description	Byte (Value)
Value	Data[4, 5] = Logical Node ID Data[6] = Node Type Data[7] = SK_LINK_UP/DOWN/NA

RequestedMatrixInfoTLV (0x8011)

Description	Byte (Value)
Tag	Data[0, 1] = 0x8011 (RequestedMatrixInfoTLV)
Length	Data[2, 3] = 0x000D
Value	Data[4, 5] = Logical Node ID Data[6, 9] = Matrix IP Data[10] = SystemType Data[11] = MatrixSide Data[12] = MatrixState Data[13] = AdjMatrixState Data[14] = StatusBits Data[15, 16] = ExcelPort

RequestedDeviceServerInfoTLV (0x8012)

Description	Byte (Value)
Tag	Data[0, 1] = 0x8012 (RequestedDeviceServerInfoTLV)
Length	Data[2, 3] = 0x000D
Value	Data[4, 5] = Logical Node ID Data[6, 9] = Matrix IP Data[10, 11] = IP Signaling Series 3 status Data[12] = SystemType Data[13] = IP Signaling Series 3 Card Type Data[14] = State Data[15] = Adjacent State Data[16] = StatusBits Data[17] = Reserved

RequestedKnownNodeInfoTLV (0x8013)

Description	Byte (Value)
Tag	Data[0, 1] = 0x8013 (RequestedKnownNodeInfoTLV)
Length	Data[2, 3] = 0x0000
Value	Data[4, 5] = Logical Node ID Data[6, 9] = IP Address

Local End Release Mode Configure 0x0021

SwitchKit Name	LocalReleaseConfig
Type	EXS API and SwitchKit API message
Description	<p>Local End Release Mode Configure 0x0021</p> <p>This message sets a channel's local end release mode, which determines whether a channel is released or parked when the other end of a connection releases.</p>
Sent by	Host

SwitchKit Code Configuration

```
LocalReleaseConfig (
    Node = Integer,
    Range = StartSpan : StartChannel - EndSpan :
    EndChannel,
    ReleaseMode = Integer);
```

C Structure

```
typedef struct {
    BaseFields Base;
    unsigned short StartSpan;
    UBYTE StartChannel;
    unsigned short EndSpan;
    UBYTE EndChannel;
    UBYTE ReleaseMode;
} XL_LocalReleaseConfig;
```

C++ Class

```
class XLC_LocalReleaseConfig : public
    XLC_ChanRangeMessage {
public:
    unsigned short getStartSpan() const;
    void setStartSpan(unsigned short x);
    UBYTE getStartChannel() const;
    void setStartChannel(UBYTE x) ;
    unsigned short getEndSpan() const;
    void setEndSpan(unsigned short x);
    UBYTE getEndChannel() const;
    void setEndChannel(UBYTE x);
    UBYTE getReleaseMode() const;
    void setReleaseMode(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0021)	3, 4	Message Type (0x0021)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u> Address Method 0x01 - Range of AEs	8, 9	Status (MSB, LSB)
	Number of AEs to follow 0x02	10	Checksum
	AEs 0x0D Channel (Starting) 0x0D Channel (Ending)		
:	Release Mode 0x01 Park Channel 0x02 Release Channel (Default) 0x03* Release Channel with Host Notify 0x04* Park Channel with Host Notify *These options work with T1 E&M Wink Start only. On Channels set with the Host Notify option, the Host receives the call processing event, <i>Receive Idle</i> . This option is valid for E&M trunk-type channels only.		
:	Checksum		

Loop Back Configure/Query 0x0091

SwitchKit Name LoopBackConfig

Type EXS API and SwitchKit API message

Description **Loop Back Configure/Query 0x0091**

This message allows you to enable or disable loopback diagnostics on the E-ONE, DS3, and T-ONE line cards. Also, this message will query the loopback mode configured on the card.

Example: The span must be out of service before you send it this message.

Sent By Host

SwitchKit Code **Configuration**

```
LoopBackConfig (
    Node = integer,
    Span = integer,
    LoopbackType = integer);
```

C Structure

```
typedef struct {
    unsigned short Span;
    UBYTE LoopbackType;
} XL_LoopBackConfig;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    unsigned short Span;
    UBYTE LoopbackType;
    UBYTE LoopbackStatus;
} XL_LoopBackConfigAck;
```

C++ Class

```
class XLC_LoopBackConfig : public XLC_SpanMessage {
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getLoopbackType() const;
    void setLoopbackType(UBYTE x);
};
```

C++ Class Response

```
class XLC_LoopBackConfigAck : public
    XLC_AcknowledgeMessage {
```

```

public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getLoopbackType() const;
    void setLoopbackType(UBYTE x);
    UBYTE getLoopbackStatus() const;
    void setLoopbackStatus(UBYTE x);
};
    
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x000E)
3, 4	Message Type (0x0091)	3, 4	Message Type (0x0091)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs Number of AEs to follow AE 0x0C Logical Span	8, 9	Status MSB, LSB Fields after the response status exist only if the response status is a positive acknowledgment.
:	Action This field indicates the loopback mode of the span. 0x00 Disable Local Loopback 0x01 Enable Local Loopback 0x02 Cancel Remote Loopback (DS3 only) 0x03 Request Remote Loopback (DS3 only) 0xFF Query Loopback	10	AIB (Individual AEs) Same as message
:		:	Data For Actions other than Query Loopback, this field is 0x00. For Action of Query Loopback (0xFF): Data [0] Action 0x00 Disable Local Loopback 0x01 Enable Local Loopback 0x02 Cancel Remote Loopback (DS3 only) 0x03 Request Remote Loopback (DS3 only) Data [1] Loopback Status (for DS3 only) 0x00 Remote Loopback Cancelled 0x01 Remote Loopback Requested
:	Checksum	:	Checksum

Loop Timing Configure 0x004A

SwitchKit Name	LoopTimingConfig
Type	EXS API and SwitchKit API message
Description	Loop Timing Configure 0x004A This message sets or clears loop timing sources.

NOTE: This message not supported on the DS3 card because it is not intended to be a loop timing source.

Sent by Host

Example Message (Socket Log Output in SwitchKit) In the following socket log output/example message, the host clears loop timing on Slot 01, Physical Span 01:

```
00 0D 00 4A 00 00 FF 00 01 13 02 01 01 01 02
```

SwitchKit Code Configuration

```
LoopTimingConfig
Node = integer,
Slot = integer,
Offset = integer,
Type = integer,
Action = integer);
```

C Structure

```
typedef struct {
    UBYTE Slot;
    UBYTE Offset;
    UBYTE Type;
    UBYTE Action;
} XL_LoopTimingConfig;
```

C++ Class

```
class XLC_LoopTimingConfig : public XLC_SlotMessage {
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getOffset() const;
    void setOffset(UBYTE x);
    UBYTE getType() const;
    void setType(UBYTE x);
    UBYTE getAction() const;
    void setAction(UBYTE x);
};
```

EXS API Hex Format

MESSAGE		RESPONSE	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x004A)	3, 4	Message Type (0x004A)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs	8, 9	Status (MSB, LSB)
	Number of AEs to follow		
	AE 0x13 Physical Span		
:	Loop Timing Type 0x01 Primary 0x02 Secondary		
:	Action 0x01 Set Loop Timing 0x02 Clear Loop Timing		
:	Checksum	10	Checksum

Matrix Configure 0x007D

SwitchKit Name	DeviceServerConfig
Type	EXS API and SwitchKit API message
Description	<p>Matrix Configure 0x007D</p> <p>This message informs a CSP Matrix Series 3 Card about an IP Signaling Series 3 card. It sets up all of the information on the CSP Matrix Series 3 Card side to allow for the connection to the IP Signaling Series 3 card.</p> <p>NOTE: This message supports a feature of an Dialogic product other than SwitchKit. For that reason, the message is not to be used in a SwitchKit only environment.</p>

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE ConfigType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_DeviceServerConfig;
```

C++ Class

```
class XLC_DeviceServerConfig : public XLC_OutboundMessage
{
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    UBYTE getConfigType() const;
    void setConfigType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

C Structure

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE ConfigType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_DeviceServerConfig;
```

C++ Class

```
class XLC_DeviceServerConfig : public XLC_OutboundMessage
{
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    UBYTE getConfigType() const;
    void setConfigType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE		RESPONSE	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x007D)	3, 4	Message Type (0x007D)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Sequence Number (same as message)
7	Logical Node ID	7	Logical Node ID
:	<u>AIBs</u> Address Method 0x00 - Individual AEs <hr/> Number of AEs to follow <hr/> AEs 0x00 00 Null (For when the IP Signaling Series 3 card ID is not yet configured. Note that this AIB is different from the four-byte null AIB used for SIP). 0x73 IP Signaling Series 3 Card ID (For when the IP Signaling Series 3 card ID is configured).	8, 9	Status MSB, LSB

Matrix Configure 0x007D

:	Configure Object Type 0x20 TLVs This field describes the type of the object the configuration is meant for. Currently the only Object Type is TLVs.	:	Checksum
:	Number of TLVs		
:	TLV 0x0001 Add IP Signaling Series 3 Card 0x0002 Remove IP Signaling Series 3 Card 0x0003 Protocol Type 0x0004 Configuration Tag 0x0005 IP Signaling Series 3 Card IP Address 0x0007 IP Signaling Series 3 Card Service State 0x0008 IP Signaling Series 3 Card Compatibility Tag 0x0009 Poll Interval 0x0009 Terminating Span/Channel, Single Address 0x000B V5 Cpath Assignment 0x000F IP Signaling Series 3 Card Interface Support		
:	Checksum		

Matrix Query 0x0097

SwitchKit Name DeviceServerQuery

Type EXS API and SwitchKit API message

Description **Matrix Query 0x0097**

This message is sent by the Host to query details of set configuration data and the status of the IP Signaling Series 3 card interface.

NOTE: This message supports a feature of a Dialogic product other than SwitchKit. For that reason, the message is **not to be used in a SwitchKit only environment**.

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE QueryType;
    UBYTE Reserved;
} XL_DeviceServerQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE TLVCount;
    UBYTE Data[250];
} XL_DeviceServerQueryAck;
```

C++ Class

```
class XLC_DeviceServerConfig : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    UBYTE getConfigType() const;
    void setConfigType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

C++ Class Response

```
class XLC_DeviceServerQueryAck : public XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
```

```

void setStatus(unsigned short x);
UBYTE getTLVCount() const;
void setTLVCount(UBYTE x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};
    
```

EXS API Hex Format

MESSAGE		RESPONSE	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x0097)	3, 4	Message Type (0x0097)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Sequence Number (same as message)
7	Logical Node ID	7	Logical Node ID
:	AIB	8, 9	Status MSB, LSB
	Address Method 0x00 - Individual AEs		
	Number of AEs to follow	10	Number of TLVs to follow
AEs 0x00 00 Null 0x73 IP Signaling Series 3 Card ID			

Matrix Query 0x0097

:	<p>Query Type</p> <p>0x01 Query List of IP Signaling Series 3 Cards Configured (NOTE: AIB is Null AIB)</p> <p>0x02 Query IP Signaling Series 3 Card Basic Information This returns for the IP Signaling Series 3 Card: Card Type Card Version Card Operational State Card Redundancy Status Card Configuration Tag Card General Data</p> <p>0x03 Query IP Signaling Series 3 Card General Information</p> <p>0x04 Query Gateways on this IP Signaling Card Series 3</p>	11	<p>TLV</p> <p>0x0001 Add IP Signaling Series 3 Card</p> <p>0x0005 IP Signaling Series 3 Card IP Address</p> <p>0x0004 Configuration Tag</p> <p>0x0007 IP Signaling Series 3 Card Service State</p> <p>0x000B V5 Cpath Assignment</p> <p>0x000F IP Signaling Series 3 Card Interface Support</p>
:	Reserved (0x00)	:	<p>Response</p> <p>0x0010 Positive Ack Everything validated and the action was initiated</p> <p>0x1200 Invalid IP Signaling Series 3 Card ID Invalid span was passed from the host</p> <p>0x1201 Invalid Query Type Invalid value for the Query Type field was passed</p> <p>0x120C IP Signaling Series 3 cards are not configured</p>
:	Checksum	:	Checksum

Matrix Status Report 0x00E5

SwitchKit Name DeviceServerStatusReport

Type EXS API and SwitchKit API message

Description **Matrix Status Report 0x00E5**

This is an unsolicited message sent from the CSP Matrix Series 3 Card to the Host in different situations. This is generally caused by an IP Signaling Series 3 card interaction about which the Host requires information.

NOTE: This message supports a feature of a Dialogic product other than SwitchKit. For this reason, the message is **not to be used in a SwitchKit only environment**.

Sent by CSP

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short DeviceServer;
    UBYTE TLVCount;
    UBYTE Data[222];
} XL_DeviceServerStatusReport;
```

C++ Class

```
class XLC_DeviceServerStatusReport : public
    XLC_InboundMessage {
public:
    unsigned short getDeviceServer() const;
    void setDeviceServer(unsigned short x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00E5)	3, 4	Message Type (0x00E5)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Sequence Number (same as message)
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs	8, 9	Status MSB, LSB
	Number of AEs to follow		
	AE 0x73 IP Signaling Series 3 Card ID		
:	Number of TLVs to follow	10	Checksum
:	TLV 0x0003 Protocol Type 0x000B V5 Cpath Assignment 0x0080 IP Signaling Series 3 Card Status		
:	Checksum		

MessageWrapper

Type SwitchKit API message

Purpose Use *SK_MessageWrapper* to provide the means to tunnel through the LLC. That is, the message to be tunneled is encapsulated inside the MessageWrapper. When the LLC receives a MessageWrapper, it extracts the payload message and delivers the extracted bytes to the proper node.

Payload The payload is a message in CSP format which includes everything from the length up to, but not including, the checksum. Here is a Card Population Query as it would formatted as C-style code:

```
UBYTE CardPop[] = {00, 05, 00, 07, 00, 02, 01};
```

That is: length (00, 05), Message Type (00, 07), Reserved (00), Sequence Number (02), and LNI(01).

The only change LLC will make is to change the Sequence Number to a suitable value following the same rules as any other message. (In-use Sequence Numbers must be unique per message type per device.)

Tunneling through the LLC Message Tunneling is a means of sending a message through the LLC to a device, without LLC attempting to process the message. A message enters the LLC and is delivered directly to the target node. When and if an acknowledgement is received, the LLC delivers to the originating application. Since LLC will not have to examine MessageWrappers for any special processing, the delivery time improves. Occasionally, a new message type is added to the CSP. Since the LLC does limited processing of tunneled messages, it doesn't need to understand the message in order to send it.

Sent by Applications

Arguments The following table shows the arguments you can change:

Parameter	Description
WaitForAck:	Set this field to a non-zero value if an acknowledgement is expected for this message. Otherwise, set it to zero.
PayloadTimeout	If WaitForAck is non-zero, then LLC will wait at least PayloadTimeout seconds for an acknowledgment.
TargetNodeId	The Logical Node ID of the device using this message.
PayloadSize	The payload size in bytes.
Payload	A properly formatted CSP format message including everything from the length up to, but not including, the checksum.

Status Argument

The following are values that can be returned in the Status argument of *MessageWrapperAck*:

Value	Description
TargetNodeId	The Logical Node ID of the device using this message.
PayloadSize	The payload size in bytes. If this is a NACK, then PayloadSize is zero.
Payload	The response from the CSP to the tunneled message.
0x10	If WaitForAck was non-zero, then this response value means that an ack was received within the specified timeout period. If WaitForAck was zero (do not wait) then 0x10 means the message was delivered to the device. A positive response does not imply that the tunneled message was successful, only that it was delivered successfully. The payload may contain a NACK.
SK_INVALID_NODE	TargetNodeId does not exist or is not LINK_UP
SK_NO_ACK_FROM_SWITCH	Timeout occurred while LLC was waiting for an acknowledgment to the tunneled message.

C Structure typedef struct {

```

        BaseFields Base;
        unsigned short TargetNodeId;
        UBYTE WaitForAck;
        UBYTE PayloadTimeout;
        unsigned short PayloadSize;
        UBYTE Payload[247];
    } XL_MessageWrapper;

```

C Structure Response

```

C-Structure
typedef struct {
    BaseFields Base;
    unsigned short Status;
    UBYTE reserved6[13];
    unsigned short TargetNodeId;
    unsigned short PayloadSize;
    UBYTE Payload[247];
} XL_MessageWrapperAck;

```

C++ Class

```

class XLC_MessageWrapper: public XLC_OutboundMessage {
    unsigned short getTargetNodeId() const;
    void setTargetNodeId(unsigned short x);
    UBYTE getWaitForAck() const;
    void setWaitForAck(UBYTE x);
    UBYTE getPayloadTimeout() const;
    void setPayloadTimeout(UBYTE x);
    unsigned short getPayloadSize() const;
    void setPayloadSize(unsigned short x);
    const UBYTE *getPayload() const;
    UBYTE *getPayload();
    void setPayload(UBYTE *x);
};

```

C++ Class Response

```

class XLC_MessageWrapperAck: public
    XLC_AcknowledgeMessage {
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    unsigned short getTargetNodeId() const;
    void setTargetNodeId(unsigned short x);
    unsigned short getPayloadSize() const;
    void setPayloadSize(unsigned short x);
    const UBYTE *getPayload() const;
    UBYTE *getPayload();
    void setPayload(UBYTE *x);
};

```

Limitations There is no way to tunnel to an application an unsolicited CSP message. The application must continue to register to receive these.

socket.log In the socket.log, messages that have been tunneled will be displayed with an asterisk preceding the bytes.

Example of an outbound message:

```
Apr 01 2002 13:44:22 H->X2[10.10.158.16] * : 01 01 00 00
    fe 00 02 52 02 00 02 53 04 00 20 00 01 02
```

Example of an inbound message:

```
Apr 01 2002 13:44:22 X2[10.10.158.16]->H * : 01 01 00 00
    fe 00 10 00 02 52 02 00 02 53 04 00 20 00 01 07 00 05
    00 02 00 03 00 08 00 02 01 01 00 07 00 02 00 01 00 0a
    00 04 00 05 00 05 00 0c 00 06 0a 0a 9e 07 31 51 00 0d
    00 06 00 00 00 00 00 00 00 0e 00 02 00 01
```

LLC message validation

The LLC will make no attempt to validate the contents of the payload. Since the CSP will not NACK ill formed messages (it generates an alarm instead) the likely response will be SK_NO_ACK_FROM_SWITCH.

Example Code This code example sends an IP Signaling Series 3 Card Query message (0x101) to a device server with a node id of 0x02:

```

UBYTE sqbuf[] = {
    0x00, 0x13, //length
    0x01, 0x01, // tag
    0x00, //reserved
    0x00, // seq num
    0xfe, // ELNI
    0x00, // address method
    0x02, // num address elements
    0x52, 0x02, 0x00, 0x02, // ELNI
    0x53, 0x04, 0x00, 0x20, 0x00, 0x01, // ObInstId
    0x02, // query type
    0x00 //reserved
};

```

```

XLC_MessageWrapper sq(sizeof(sqbuf)+0x20);

```

```

sq.setPayloadSize(sizeof(sqbuf));
memcpy(sq.getPayload(), sqbuf, sizeof(sqbuf));
sq.setTargetNodeId(0x02);
sq.setWaitForAck(1);
sq.setPayloadTimeout(0x10);
sq.send();

```

This code example is from the handler function for this message:

```

CASEC_MessageWrapperAck(mwa)
{
    cout << "ACK: " << hex << mwa->getStatus() << endl;
    cout << "    Payload size: " << mwa->getPayloadSize() << endl;
    cout << "    Payload: ";

    if (mwa->getStatus() == 0x10)
    {
        for (int x = 0; x<mwa->getPayloadSize(); x++)
        {
            cout << hex << (int)mwa->getPayload()[x] << " ";
        }
        cout << endl;
    }
    return OK;
}

```

MonitorChannel

Type	SwitchKit API message
Description	This message is used by the function sk_monitorChannel() in case an application makes a monitoring call.
Sent by	Function sk_monitorChannel()

Multi-Host Configure 0x00E9

SwitchKit Name MultiHostConfig

Type EXS API and SwitchKit API message

Description **Multi-Host Configure 0x00E9**

Use this message to register a host to receive or not receive specific API messages, as well as the PPL components for which *PPL Event Indication* messages are sent by the CSP. You also use this message to query the registration status for a specific host.

Note the following:

- You can only send one ICB subtype per message.
- Multi-Host Control is disabled by default. You enable it with Action ICB Subtype 0x33.

Definitions

Hard Register - CSP-initiated messages or *PPL Event Indications* messages will be received by the host regardless if any other hosts are receiving it.

Soft Register - CSP-initiated messages or *PPL Event Indications* messages will be received by the host only if the messages are not destined for any other host, either by none registering for them or by the registered host(s) not having socket connectivity.

De-register - CSP-initiated messages or *PPL Event Indications* messages will not be received by the host under any circumstances.

PPL-Component IDs - PPL-component IDs are a sub-field in the *PPL Event Indication* message. Registration for these IDs will cause the *PPL Event Indication* message with this ID as a sub-field to be sent to the registered host. To receive all *PPL Event Indication* messages, a host should register for the *PPL Event Indication* message itself, and not use the PPL-component ID registration.

Sent by Host

SwitchKit Code **Configuration**

```
MultiHostConfig (
    Node = integer,
    Data = byte array);
```

C Structure

```
typedef struct {
    UBYTE Data[253];
} XL_MultiHostConfig;
```

C Structure Response

```
typedef struct {
    tus;
    UBYTE Data[251];
} XL_MultiHostConfigAck;
```

C++ Class

```
class XLC_MultiHostConfig : public XLC_OutboundMessage {
public:
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

C++ Class Response

```
class XLC_MultiHostConfigAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

Overview of message

The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00E9)	3, 4	Message Type (0x00E9)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	ICB	8, 9	Status MSB, LSB
:	:	10	Data
:	Checksum	:	Checksum

EXS API Hex Format - Detailed

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00E9)	3, 4	Message Type (0x00E9)

Multi-Host Configure 0x00E9

5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Number of ICBs to follow (Ignore this field if no ICBs in message.)	8, 9	Status MSB, LSB
:	<p>ICB All ICBs valid for this message are listed below.</p> <p>0x01 Action ICBs 0x33 Enable Multi-Host * 0x34 Disable Multi-Host * 0x35 Query Host Message Registration 0x36 Query Host Connections * 0x37 Query Multi-Host Status * 0x38 Query Hard-registered PPL Components 0x39 Query Soft-registered PPL Components</p> <p>* The ICB has no data, so you will always use 0x00 for the value of the Data Length field.</p> <p>0x02 Data ICBs 0x47 Enable Switch-Initiated Messages 0x48 Disable Switch-Initiated Messages 0x49 Enable PPL Event Indication Messages 0x4A Disable PPL Event Indication Messages 0x4B Deregister Switch-Initiated Messages 0x4C Deregister PPL Components 0x50 Hard Register All Switch-Initiated Messages 0x51 Soft Register All Switch-Initiated Messages 0x52 Deregister All Switch-Initiated Messages 0x53 Hard Register All PPL Component IDs 0x54 Soft Register All PPL Component IDs 0x55 Deregister All PPL Component IDs</p>	10	<p>Data [0]</p> <p>0x33 Enable Multi-Host (Contains no data) 0x34 Disable Multi-Host (Contains no data)</p> <p>0x35 Query Host Message Registration Data[0-3] Host ID (4 bytes) Data[4-5] Number of Messages (2 bytes) Data[6-7] Message Type 1 (2 bytes) Data[8-9] Message Type 2 (2 bytes) : : Data[:] Message Type N (2 bytes)</p> <p>0x36 Query Host Connections Data[0-3] Default Host ID (4 bytes) (00 00 00 00 if not connected) Data[4-5] Number of hosts connected (2 bytes) Data[6-9] Host ID 1 (4 bytes; first connected additional host) Data[10-13] Host ID 2 (4 bytes) : : Data[:] Host ID N (4 bytes)</p> <p>0x37 Query Multi-Host Status Data[0-3] Default Host ID (4 bytes) Data[4] Status 0x00 Disabled 0x01 Enabled</p> <p>0x38 Query PPL Components Data[0-3] Host ID (4 bytes) Data[4-5] Number of Components (2 bytes) Data[6-7] Component ID 1 (2 bytes) Data[8-9] Component ID 2 (2 bytes) : : Data[:] Component ID N (2 bytes)</p> <p>0x47 Enable CSP-Initiated Messages (Contains no data) 0x48 Disable CSP-Initiated Messages (Contains no data) 0x49 Enable <i>PPL Event Indication</i> Messages (Contains no data) 0x4A Disable <i>PPL Event Indication</i> Messages (Contains no data)</p>
:	Checksum	:	Checksum

NGA Configure 0x0130

SwitchKit Name	NGAConfigure
Type	EXS and SwitchKit API message
Description	NGA Configure 0x0130 Configures the M3UA stack and objects.
Sent by	Host application
Related API Messages	<i>NGA Service Configure (0x0160)</i> <i>NGA Configure Query (0x0131)</i> <i>NGA Service Query (0x0161)</i> <i>NGA State Notify (0x0163)</i> <i>NGA State Query (0x0162)</i>

SwitchKit Code C Structure

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_NGAConfigure;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[219];
} XL_NGAConfigureAck;
```

C++ Class

```
class XLC_NGAConfigure : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    void setAIBType(UBYTE x);
    UBYTE getAIBType();
    void setAECCount(UBYTE x);
    UBYTE getAECCount();
    void addAE(UBYTE AEPoS, UBYTE AEType, UBYTE AElen,
    UBYTE* AEvalue);
    UBYTE* getAE(int AEPoS);
```

```

    void setAIBData(UBYTE* AIBDataPtr, UBYTE
AIBDataLen);
    UBYTE* getAIBData();
    UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
}

```

C++ Class Response

```

class XLC_NGAConfigureAck : public
XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    void setAIBType(UBYTE x);
    UBYTE getAIBType();
    void setAECCount(UBYTE x);
    UBYTE getAECCount();
    void addAE(UBYTE AEPoS, UBYTE AEType, UBYTE AElen,
UBYTE* AEvalue);
    UBYTE* getAE(int AEPoS);
    void setAIBData(UBYTE* AIBDataPtr, UBYTE
AIBDataLen);
    UBYTE* getAIBData();
    ;UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData();
    UBYTE *getData();
    void setData(UBYTE *x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0130)	3, 4	Message Type (0x0130)
5, 6	Sequence Number	5, 6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB	8, 9	
	Address Method 0x03 - Item	10	Checksum
	Number of AEs to Follow		
	AEs 0x65 Global Object ID 0x67 Functional Area 0x08 SS7 Stack 0x64 Global Object Type		
:	Data Type 0x00 - TLVs		
:	Number of TLVs		
:	0xE001 Command		
:	Data TLVs		
	0x9041 Protocol Parameters		
	0x9042 Application Server		
	0x9043 Application Server Process		
	0x9044 Signaling Gateway		
	0x9045 Signaling Gateway Process		
	0x9046 Connection		
0x9052 Route Set			
0x9053 Modify Destination			
	Checksum		

NGA Configure Query 0x0131

SwitchKit Name	NGAConfigureQuery
Type	EXS and SwitchKit API message
Description	NGA Configure Query 0x0131 Queries the M3UA objects for their service state.
Sent by	Host application
Related API Messages	<i>NGA Configure (0x0130)</i> <i>NGA Service Configure (0x0160)</i> <i>NGA Service Query (0x0161)</i> <i>NGA State Notify (0x0163)</i> <i>NGA State Query (0x0162)</i>

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_NGAConfigureQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[219];
} XL_NGAConfigureQueryAck;
```

C++ Class

```
class XLC_NGAConfigureQuery : public
    XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    void setAIBType(UBYTE x);
    UBYTE getAIBType();
    void setAECCount(UBYTE x);
    UBYTE getAECCount();
    void addAE(UBYTE AEPoS, UBYTE AEType, UBYTE AElen,
        UBYTE* AEvalue);
    UBYTE* getAE(int AEPoS);
```

```

    void setAIBData(UBYTE* AIBDataPtr, UBYTE
AIBDataLen);
    UBYTE* getAIBData();
    UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};

```

C++ Class Response

```

class XLC_NGAConfigureQueryAck : public
XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    void setAIBType(UBYTE x);
    UBYTE getAIBType();
    void setAECCount(UBYTE x);
    UBYTE getAECCount();
    void addAE(UBYTE AEPoS, UBYTE AEType, UBYTE AElen,
UBYTE* AEvalue);
    UBYTE* getAE(int AEPoS);
    void setAIBData(UBYTE* AIBDataPtr, UBYTE
AIBDataLen);
    UBYTE* getAIBData();
    ;UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData();
    UBYTE *getData();
    void setData(UBYTE *x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0131)	3, 4	Message Type (0x0131)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIBs	8	Data Type 0x00 TLVs
	Address Method 0x03 - Item	9	Number of TLVs to Follow
	Number of AEs to Follow	:	TLVs 0x904A Protocol Query 0x9050 General Query Data 0x904C Application Server Process Query 0x904E Signaling Gateway Process Query 0x904D Signaling Gateway Query 0x904F Connection Query 0x9051 Application Server State Query 0xE002 Service State
	AEs 0x67 Functional Area 0x08 SS7 Stack 0x64 Global Object Type 0xFFFF - Complete list of objects		
:	Checksum	:	Checksum

NGA Service Configure 0x0160

SwitchKit Name NGAServiceConfigure

Type **NGA Service Configure 0x0160**
 EXS and SwitchKit API message

Description Changes the service state for the M3UA object.

Sent by Host application

Related API Messages

- NGA Configure (0x0130)*
- NGA Configure Query (0x0131)*
- NGA Service Query (0x0161)*
- NGA State Notify (0x0163)*
- NGA State Query (0x0162)*

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_NGAServiceConfigure;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[219];
} XL_NGAServiceConfigureAck;
```

C++ Class

```
class XLC_NGAServiceConfigure    : public
    XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    void setAIBType(UBYTE x);
    UBYTE getAIBType();
    void setAECCount(UBYTE x);
    UBYTE getAECCount();
    void addAE(UBYTE AEpos, UBYTE AEType, UBYTE AElen,
    UBYTE* AEvalue);
    UBYTE* getAE(int AEpos);
```

```

    void setAIBData(UBYTE* AIBDataPtr, UBYTE
AIBDataLen);
    UBYTE* getAIBData();
    UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
}

```

C++ Class Response

```

class XLC_NGAServiceConfigureAck : public
XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    void setAIBType(UBYTE x);
    UBYTE getAIBType();
    void setAECCount(UBYTE x);
    UBYTE getAECCount();
    void addAE(UBYTE AEPoS, UBYTE AEType, UBYTE AElen,
UBYTE* AEvalue);
    UBYTE* getAE(int AEPoS);
    void setAIBData(UBYTE* AIBDataPtr, UBYTE
AIBDataLen);
    UBYTE* getAIBData();
    ;UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData();
    UBYTE *getData();
    void setData(UBYTE *x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0160)	3, 4	Message Type (0x0160)
5, 6	Sequence Number	5, 6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB	8	Checksum
	Address Method 0x03 - Item		
	Number of AEs to Follow		
	AEs 0x67 Functional Area 0x08 SS7 Stack 0x64 Global Object Type		
:	Data Type 0x00 - TLVs		
:	Number of TLVs		
:	0xE001 Command		
:	Data TLV 0x9048 Application Server In Service		
:	Checksum		

NGA Service Query 0x0161

SwitchKit Name NGAServiceQuery

Type EXS and SwitchKit API message

Description **NGA Service Query 0x0161**

Queries the objects for their service state. The service state is the state that the user put the object into. For example, if the user put a link in service but the link is not physically connected, the service state is *In Service*

Sent by Host application

Related API Messages

- NGA Configure (0x0130)*
- NGA Service Configure (0x0160)*
- NGA Configure Query (0x0131)*
- NGA State Query (0x0162)*
- NGA State Notify (0x0163)*

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_NGAServiceQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[219];
} XL_NGAServiceQueryAck;
```

C++ Class

```
class XLC_NGAServiceQuery : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    void setAIBType(UBYTE x);
    UBYTE getAIBType();
```

```

void setAECCount(UBYTE x);
UBYTE getAECCount();
void addAE(UBYTE AEPoS, UBYTE AEType, UBYTE AElen,
UBYTE* AEvalue);
UBYTE* getAE(int AEPoS);
void setAIBData(UBYTE* AIBDataPtr, UBYTE
AIBDataLen);
UBYTE* getAIBData();
UBYTE getDataType() const;
void setDataType(UBYTE x);
UBYTE getTLVCount() const;
void setTLVCount(UBYTE x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
}

```

C++ Class Response

```
class XLC_NGAServiceQueryAck : public
```

```
XLC_AcknowledgeMessage {
```

```
public:
```

```

unsigned short getStatus() const;
void setStatus(unsigned short x);
const UBYTE *getAddrInfo() const;
UBYTE *getAddrInfo();
void setAddrInfo(UBYTE *x);
void setAIBType(UBYTE x);
UBYTE getAIBType();
void setAECCount(UBYTE x);
UBYTE getAECCount();
void addAE(UBYTE AEPoS, UBYTE AEType, UBYTE AElen,
UBYTE* AEvalue);
UBYTE* getAE(int AEPoS);
void setAIBData(UBYTE* AIBDataPtr, UBYTE
AIBDataLen);
UBYTE* getAIBData();
;UBYTE getDataType() const;
void setDataType(UBYTE x);
UBYTE getTLVCount() const;
void setTLVCount(UBYTE x);
const UBYTE *getData();
UBYTE *getData();
void setData(UBYTE *x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0161)	3, 4	Message Type (0x0161)
5, 6	Sequence Number	5, 6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB	8	Data Type 0x00 TLVs
	Address Method 0x03 - Item	9	Number of TLVs to Follow
	Number of AEs to Follow	10, 11	TLVs 0x9051 Application Server State Query
	AEs 0x67 Functional Area 0x08 SS7 Stack 0x64 Global Object Type	:	Checksum
:	Checksum		

NGA State Query 0x0162

SwitchKit Name NGAStateQuery

Type EXS and SwitchKit API message

Description **NGA State Query 0x0162**

Queries the M3UA objects for their actual state status. Do not confuse with the Service State Query. The following example shows the difference. If the user puts a link in service but the link is not physically connected, the service state is *In Service* but the state status is *Out of Service*.

Sent by Host application

Related API Messages *NGA Configure (0x0130)*
NGA Service Configure (0x0160)
NGA Configure Query (0x0131)
NGA Service Query (0x0161)
NGA State Notify (0x0163)

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_NGAStateQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[219];
} XL_NGAStateQueryAck;
```

C++ Class

```
class XLC_NGAStateQuery : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    void setAIBType(UBYTE x);
    UBYTE getAIBType();
```

```

void setAECCount(UBYTE x);
UBYTE getAECCount();
void addAE(UBYTE AEpos, UBYTE Aetype, UBYTE AElen,
UBYTE* AEvalue);
UBYTE* getAE(int AEpos);
void setAIBData(UBYTE* AIBDataPtr, UBYTE
AIBDataLen);
UBYTE* getAIBData();
UBYTE getDataType() const;
void setData(UBYTE x);
UBYTE getTLVCount() const;
void setTLVCount(UBYTE x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
}

```

C++ Class Response

```

class XLC_NGASStateQueryAck : public
XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    void setAIBType(UBYTE x);
    UBYTE getAIBType();
    void setAECCount(UBYTE x);
    UBYTE getAECCount();
    void addAE(UBYTE AEpos, UBYTE Aetype, UBYTE AElen,
    UBYTE* AEvalue);
    UBYTE* getAE(int AEpos);
    void setAIBData(UBYTE* AIBDataPtr, UBYTE
    AIBDataLen);
    UBYTE* getAIBData();
    ;UBYTE getDataType() const;
    void setData(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData();
    UBYTE *getData();
    void setData(UBYTE *x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0162)	3, 4	Message Type (0x0162)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB	8	Data Type 0x00 TLVs
	Address Method 0x03 - Item	9	Number of TLVs to Follow
	Number of AEs to Follow	10, 11	TLVs 0xE002 Service State
	AEs 0x67 Functional Area 0x08 SS7 Stack 0x64 Global Object Type	:	Checksum
Checksum	:		

NGA State Notify 0x0163

SwitchKit Name	NGAStateNotify
Type	EXS and SwitchKit API message
Description	<p>NGA State Notify 0x0163</p> <p>The M3UA object uses this message to notify the host about changes in object status.</p>
Sent by	CSP
Related API Messages	<p><i>NGA Configure (0x0130)</i></p> <p><i>NGA Configure Query (0x0131)</i></p> <p><i>NGA Service Configure (0x0160)</i></p> <p><i>NGA Service Query (0x0161)</i></p> <p><i>NGA State Query (0x0162)</i></p>

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_NGAStateNotify;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[219];
} XL_NGAStateNotifyAck;
```

C++ Class

```
class XLC_NGAStateNotify : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    void setAIBType(UBYTE x);
    UBYTE getAIBType();
    void setAECCount(UBYTE x);
    UBYTE getAECCount();
    void addAE(UBYTE AEPoS, UBYTE AEType, UBYTE AElen,
    UBYTE* AEvalue);
    UBYTE* getAE(int AEPoS);
```

```

    void setAIBData(UBYTE* AIBDataPtr, UBYTE
AIBDataLen);
    UBYTE* getAIBData();
    UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};

```

C++ Class Response

```

class XLC_NGASStateNotifyAck : public
XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    void setAIBType(UBYTE x);
    UBYTE getAIBType();
    void setAECCount(UBYTE x);
    UBYTE getAECCount();
    void addAE(UBYTE AEPoS, UBYTE AEType, UBYTE AElen,
UBYTE* AEvalue);
    UBYTE* getAE(int AEPoS);
    void setAIBData(UBYTE* AIBDataPtr, UBYTE
AIBDataLen);
    UBYTE* getAIBData();
    ;UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData();
    UBYTE *getData();
    void setData(UBYTE *x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0163)	3, 4	Message Type (0x0163)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB	8	Data Type 0x00 TLVs
	Address Method 0x03 - Item	9	Number of TLVs to Follow
	Number of AEs to Follow	:	TLVs 0x9051 Application Server State Query 0xE002 Service State
	AEs 0x67 Functional Area 0x7F SIP Stack ID 0x64 Global Object Type		
:	Checksum	:	Checksum

Node Status Query 0x006F

SwitchKit Name	NodeStatusQuery
Type	EXS API and SwitchKit API message
Description	<p>Node Status Query 0x006F</p> <p>Use this message to query the status of a CSP node in a multi-node system. The status is reported in the response.</p>
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {
    } XL_NodeStatusQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    int PhysicalNode;
    UBYTE LogicalNode;
    UBYTE HostNode;
    UBYTE NodeStatus;
    } XL_NodeStatusQueryAck;
```

C++ Class

```
class XLC_NodeStatusQuery : public XLC_OutboundMessage {
public:
    };
```

C++ Class Response

```
class XLC_NodeStatusQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    int getPhysicalNode() const;
    void setPhysicalNode(int x);
    UBYTE getLogicalNode() const;
    void setLogicalNode(UBYTE x);
    UBYTE getHostNode() const;
    void setHostNode(UBYTE x);
    UBYTE getNodeStatus() const;
    void setNodeStatus(UBYTE x);
    };
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x006F)	3, 4	Message Type (0x006F)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u> Address Method 0x00 - Individual AEs	8, 9	Status MSB, LSB
	Number of AEs to follow		
	AE		
:	Checksum	10	AIB Same as message
:	Host Node ID The value here is the same as the Logical Node ID field in the header. The system uses the value 0xF0 for a host node that does not have a logical node ID.		
:	Node Status This field indicates the current status of the node. 0x00 Node Initialized 0x01 Reassigning Host Node – A new host is indicated in the Host Node field 0x04 Multi Node System – ISUP Remote Control 0xFF Node Requires Logical Node ID		
:	Checksum		

Node Status Report 0x0070

SwitchKit Name	NodeStatusReport
Type	EXS API and SwitchKit API message
Description	Node Status Report 0x0070 A CSP in a multi-node system sends this message to the host after the node reboots or its status changes.
Sent by	CSP

SwitchKit Code **C Structure**

```
typedef struct {  
    int PhysicalNode;  
    UBYTE LogicalNode;  
    UBYTE HostNode;  
    UBYTE NodeStatus;  
} XL_NodeStatusReport;
```

C++ Class

```
class XLC_NodeStatusReport : public XLC_InboundMessage {  
public:  
    int getPhysicalNode() const;  
    void setPhysicalNode(int x);  
    UBYTE getLogicalNode() const;  
    void setLogicalNode(UBYTE x);  
    UBYTE getHostNode() const;  
    void setHostNode(UBYTE x);  
    UBYTE getNodeStatus() const;  
    void setNodeStatus(UBYTE x);  
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length, MSB (0x0005)
3, 4	Message Type (0x0070)	3, 4	Message Type (0x0070)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u> Address Method 0x00 - Individual AEs Number of AEs to follow AE 0x11 Logical/Physical Span	8	Checksum
:	Host Node ID The value is the same as the <i>Logical Node ID</i> field in the header. The system uses the value 0xF0 for a host node that does not have a logical node ID.		
:	Node Status This field indicates the current status of the node. 0x00 Node Initialized 0x04 Multi Node System – ISUP Remote Control 0xFF Node Requires Logical Node ID		
:	Checksum		

Outputpulse Digits 0x0020

SwitchKit Name OutputpulseDigits

Type EXS API and SwitchKit API message

Description **Outputpulse Digits 0x0020**

This message instructs the CSP to outputpulse up to two strings of digits to the specified channel. The strings may be in BCD or non-BCD format.

Using non-BCD format allows the host to insert silence between the outputpulsing of digits by inserting delay digits (0x10) within the digit string. When a delay digit is encountered in a digit string, the DSP providing the outputpulsing will transmit silence for the delay duration. The delay duration is configured by the host in this message. Delay digits may be grouped to provide dynamic delay durations.

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short Span;
    UBYTE Channel;
    UBYTE SignalType;
    UBYTE StringCount;
    UBYTE StringFormat;
    UBYTE StringMode;
    unsigned short FirstDigitDuration;
    unsigned short DigitDuration;
    unsigned short InterDigitDuration;
    unsigned short DelayDuration;
    UBYTE GenerateEventFlag;
    UBYTE StringsData[210];
} XL_OutputpulseDigits;
```

C++ Class

```
class XLC_OutputpulseDigits : public XLC_OneChannelOutbound
{
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getSignalType() const;
    void setSignalType(UBYTE x);
    UBYTE getStringCount() const;
    void setStringCount(UBYTE x);
    UBYTE getStringFormat() const;
    void setStringFormat(UBYTE x);
    UBYTE getStringMode() const;
    void setStringMode(UBYTE x);
    unsigned short getFirstDigitDuration() const;
```

```

void setFirstDigitDuration(unsigned short x);
unsigned short getDigitDuration() const;
void setDigitDuration(unsigned short x);
unsigned short getInterDigitDuration() const;
void setInterDigitDuration(unsigned short x);
unsigned short getDelayDuration() const;
void setDelayDuration(unsigned short x);
UBYTE getGenerateEventFlag() const;
void setGenerateEventFlag(UBYTE x);
const UBYTE *getStringsData() const ;
UBYTE *getStringsData();
void setStringsData(UBYTE *x);
};

```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0020)	3, 4	Message Type (0x0020)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB (starting with byte 0)	8, 9	Status MSB, LSB
:	:	10, 11	State If the preceding Status field is a positive acknowledgement, this field does not apply.
:	Signal Type		
		12	Checksum
:	String Count		
:	String Format		
:	String Mode		
:	First Digit Duration MSB, LSB		
:	Digit Duration MSB, LSB		
:	Interdigit Duration MSB, LSB		
:	Delay Duration MSB, LSB		
:	Generate Event Flag		
:	String 1 Digit Count		
:	String 1 Digits		
:	:		
:	String 2 Digit Count		
:	String 2 Digits		
:	:		
:	Checksum		

**EXS API Hex Format -
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0020)	3, 4	Message Type (0x0020)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs Number of AEs to follow AE 0x0D Channel	8, 9	Status (MSB, LSB)
:	Signal Type 0x01 DTMF 0x02 MFR1	10	Checksum
:	String Count Number of digit strings to be outpulsed (0x01–0x02)		
:	String Format 0x00 BCD (two digits per byte, delay digits not valid) 0x01 Non-BCD (one digit per byte, delay digits valid)		
:	String Mode This field is valid only if signal type is MFR1. (Set to 0x00 for DTMF) 0x00 The CSP will insert KP and ST framing digits. 0x01 The host will insert framing digits.		
:	First Digit Duration MSB, LSB This field is valid only if signal type is MFR1. (Set to 0x00 for DTMF) A two-byte field identifying the length of time to outpulse the KP signal. The typical duration is 100 ms (0x000A).		
:	Digit Duration MSB, LSB A two-byte field identifying the length of time to outpulse the digits. The typical duration is 80 ms (0x0008).		
:	Interdigit Duration MSB, LSB A two-byte field identifying the length of time to wait between each outpulsed digit. The typical duration is 80 ms (0x0008).		
:	Delay Duration MSB, LSB A two-byte field identifying the length of time to wait when the delay digit (0x10) is encountered in the digit string. The typical duration is 40 ms (0x0004); it should not be less than 20 ms (0x0002). When determining the delay duration, keep in mind that the inter-digit duration will apply between digit delays.		
:	Generate Event Flag 0x00 Do not inform host when outpulsing complete 0x01 Inform host when outpulsing complete with a Call Processing Event message specifying event type as “outpulsing complete.”		

Outpulse Digits 0x0020

:	String 1 Digit Count Number of digits in string to follow (0x01–0x64). This message allows for 100 bytes of digit data which means: if the host specifies string format as BCD, it may outpulse two BCD strings of length 100 each, and if the host specifies string format as Non-BCD, it may outpulse one string of 100 digits, or two strings with the sum of digits in both strings equal to 100.
:	String 1 Digits (See table below)
:	:
:	String 2 Digit Count
:	String 2 Digits
:	:
	Refer to <i>Interworking TLVs (4-4)</i> if you have enabled interworking from the <i>VoIP Protocol Configure 0x00EE</i> message.
:	Checksum

String NN Digits

Call State Values in the More Status Field

Call State	Pre-5.3 Values	CSP Call Control CH Values
Out-of-Service	0x00 Out-of-Service	0x00 Out-of-Service
Idle	0x01 Reserved	0x03 In Service Idle
Connect Wait	0x02 Reserved	0x04 Incoming Call, Wait for L5
Book Wait	0x03 Idle	0x06 Wait For CSA in Incoming Call Wait for Host State
	0x04 Connect Wait	0x01 Incoming Call
Answer Wait2	0x05 Outseize Wait	0x0E Incoming Call Alerted
Connect	0x06 Book Wait	0x07 Answered
	0x07 Connect	0x02 Wait For CSA in Incoming Call State
	0x08 Release Wait	0x0C Wait for CSA in Incoming Alerted State
	0x09 Clear Wait	0x0D Wait for CSA in Incoming in Answered State
Clear Wait	0x0A Busied Out	0x09 L3 Clear Wait
	0x0B Outseized	0x0F L3 Clear Wait
Busied Out	0x0C Hold Acknowledgment Wait	0x0A Busied out
	0x0D Answer Wait	0x10 Wait for CSA in Outgoing Cut-thru
Release With Data Wait	0x0E Answer Wait2	0x15 L5 Release Wait
Outseize Wait	0x0F Book Wait2	0x05 Outgoing Call, L3 Outseize Wait
Parked	0x10 Recall Wait	0x13 Outgoing Call Alerted
	0x11 Purge Response Wait	0x11 Wait for CSA in Outgoing Alerted State
Outseized	0x12 Purge Wait	0x0B Outgoing Call Cut-thru
	0x13 Parked	0x12 L4 Clear ACK Wait, L3 Clear Received
Release Wait	0x14 Broadcast Create Wait	0x08 L4 Clear ACK Wait, L3 Disconnect Received
	0x15 Release With Data Wait	0x14 L4 Clear ACK Wait, L5 Clear Received
		0x16 Incoming Call, L4 Clear ACK Wait (Park Processing)
		0x17 Incoming Call Alerted, L4 Clear ACK Wait (ParkProcessing)
		0x18 Answered, L4 Clear ACK Wait (Park Processing)
		0x19 Outgoing Alerted, L4 Clear ACK Wait (Park Processing)
		0x1A Outgoing Cut-thru, L4 Clear ACK Wait (Park Processing)
		0x1B Wait for CSA for Internal Routing

Digit Transmissions

The following table shows you the hexadecimal equivalents of DTMF and MFR1 digits:

DTMF Digit	Hex Value	MFR1 Digit	Hex Value
1	0x01	1	0x01
2	0x02	2	0x02
3	0x03	3	0x03
4	0x04	4	0x04
5	0x05	5	0x05
6	0x06	6	0x06
7	0x07	7	0x07
8	0x08	8	0x08
9	0x09	9	0x09
0	0x00	0	0x00
*	0x0E	KP	0x0A
#	0x0F	ST	0x0B
A	0x0A	ST1	0x0C
B	0x0B	ST2	0x0D
C	0x0C	ST3	0x0E
D	0x0D	Delay	0x10
Delay	0x10		

Outseize Control 0x002C

SwitchKit Name OutseizeControl

Type EXS API and SwitchKit API message

Description **Outseize Control 0x002C**

Use this message to control outgoing call setup during real-time call processing on the specified channel. Instructions and data are passed in the form of Information Control Blocks (ICBs).

You can preprogram a list of instructions using the *Outseize Instruction List Configure* message and initiate them by sending the *Outseize Control* message with an ICB of "Use Instruction List." Note the seize instruction cannot be pre-programmed.

An instruction that is not carried out reports a negative Acknowledgment. A positive acknowledgment is reported only if an Action ICB of "Send Host Acknowledgment" is included in the message.

Contact Dialogic Technical Support if you want to send an *Outseize Control* message greater than 512 bytes.

ISDN PRI and SS7

Outseize instructions cannot be preprogrammed for ISDN PRI or SS7 channels. You must include all outseize instructions for ISDN PRI and SS7 in the *Outseize Control* message.

For an SS7 channel, this message results in an IAM being sent to the network. Mandatory parameters must be included in an SS7 Parameters ICB.

- Related Messages**
- *Outseize Instruction List Configure* 0x002A (OutseizeInstrListConfig)
 - *Call Control Instructions Query* 0x0087 (CallControlInstructionQuery)

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short Span;
    UBYTE Channel;
    UBYTE ICBCount;
    UBYTE ICBDData[222];
} XL_OutseizeControl;
```

C Structure Response

```
typedef struct {
    unsigned short Span;
    UBYTE Channel;
    UBYTE reserved20[27];
    UBYTE ICBCount;
    UBYTE ICBDData[222];
```

```
} XLC_OutseizeControlAck;
```

C++ Class

```
class XLC_OutseizeControl : public XLC_OneChannelOutbound  
{  
public:  
    unsigned short getSpan() const;  
    void setSpan(unsigned short x);  
    UBYTE getChannel() const;  
    void setChannel(UBYTE x);  
    UBYTE getICBCount() const;  
    void setICBCount(UBYTE x);  
    const UBYTE *getICBData() const;  
    UBYTE *getICBData();  
    void setICBData(UBYTE *x);  
};
```

C++ Class Response

```
class XLC_OutseizeControlAck : public  
XLC_AcknowledgeMessage {  
public:  
    unsigned short getStatus() const;  
    void setStatus(unsigned short x);  
    const UBYTE *getData() const;  
    UBYTE *getData();  
    void setData(UBYTE *x);  
};
```

Overview of Message

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0009)
3, 4	Message Type (0x002C)	3, 4	Message Type (0x002C)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB (starting with Byte 0)	8, 9	Status (MSB, LSB)
:	ICB	10, 11	State
:	:		
:	Checksum	:	Checksum

EXS API Hex Format-Detailed

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length 0x0007 Positive Acknowledgement 0x0009 Negative Acknowledgement
3, 4	Message Type (0x002C)	3, 4	Message Type (0x002C)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID

Outseize Control 0x002C

:	<u>AIB</u> Address Method 0x00 - Individual AEs	8, 9	Status (MSB, LSB) For SSUTR2, the following NACKs apply: 0x0A Continuity Check resources not available 0x0B Incoming specialized circuit 0x0C Congestion 0x0F Blocked For ISUP, the following NACKs apply: 0x0A Continuity Recheck In Progress 0x0B CIC Hardware Remotely Blocked 0x0C CIC in Outgoing Congestion 0x0D CIC Locally Maintenance Blocked 0x0E CIC Remotely Hardware Blocked 0x0F CIC Remotely Maintenance Blocked 0x51 00 ISUP Congestion
	Number of AEs to follow	10, 11	State If the preceding Status field is a positive acknowledgement, this field does not apply.
	AE 0x0D Channel	:	Number of ICBs to follow
:	Number of ICBs to follow		ICB Subtype
		:	Checksum

Outseize Control 0x002C

:	<p>ICB</p> <p>NOTE: For an SS7 channel in ANSI ISUP CRM/CRA, the <i>Outseize Control</i> message must include an SS7 Parameters ICB. This allows a CRM message type to be part of the SS7 Parameter ICB that is sent in the <i>Outseize Control</i> message. To send an IAM message after an <i>Outseize Control</i> message with CRM, a separate <i>PPL Event Request</i> for IAM must be made.</p> <p>0x01 Action</p> <ul style="list-style-type: none"> 0x00 Null* 0x01 Scan For Wink N 0x02 Scan For ANI Request Off-hook* 0x03 Scan For Dial Tone* 0x04 Report Call Processing Event 0x05 Outpulse Stage N Address Data 0x06 Wait For Host Address Data* 0x07 Wait For Host Control* 0x08 Send Host Acknowledgment * (the ISDN PRI card ignores the Send Host Acknowledgment) 0x09 Do Call Progress Analysis (T-ONE calls only) 0x0A Seize (see Note 3)* 0x0B Use Instruction List 0x0C Reserved* 0x0D Cancel R2 Receiver* 0x0E Scan For Backward R2 Signal* 0x0F Wait For Host Control With Answer Supervision (see Note 4)* 0x10 Do Call Progress Analysis Without Line Signaling 0x11 Delay N Milliseconds <p>0x02 Data</p> <ul style="list-style-type: none"> 0x01 Stage N Address Data 0x02 Stage N Address Data 0x10 ISDN Formatted IEs 0x11 ISDN Raw IEs 0x12 SS7 Parameters 0x15 DASS2/DPNSS Raw Data 0x1C SS7 TUP Formatted Fields 0x1E Generic PPL 0x25 ISDN Segmented Message 0x66 SS7 Address Information <p>0x03 Extended Data</p> <ul style="list-style-type: none"> 0x0012 SS7 Formatted Parameters 0x0033 NPDI Universal ICB 0x001E Generic PPL <p>* The ICB has no data, so you will always use 0x00 for the value of the Data Length field.</p>
:	Checksum

Notes:

1. When the system outpulses BCD-encoded digits, a Stage N Address Data ICB must accompany the digits to define the outpulsing parameters. The position of the Data ICB is not critical.
2. The host must have a DSP configured for Call Progress Analysis for a “Do Call Progress Analysis” ICB to be invoked. For more information, refer to the message *DSP SIMM Configure* (0xC0)
3. A seize instruction, when present, must be the first Action ICB. A seize instruction cannot be included in a preprogrammed outseize instruction list.
4. If Answer Supervision Mode is configured for “Notify” or “Notify and Propagate,” the host will be informed with a Call Processing Event message of “Answer.” Note that the default Answer Supervision Mode is “Propagate.”

Outseize Instruction List Configure 0x002A

SwitchKit Name OuseizeInstrListConfig

Type EXS API and SwitchKit API message

Description **Outseize Instruction List Configure 0x002A**

Use several of these messages to preprogram a list of instructions for outseize control on CAS channels. You must send a separate *Outseize Instruction List Configure* message for each instruction that you add to the list.

Outseize instructions cannot be preprogrammed for ISDN PRI or SS7 channels. You must include all outseize instructions for ISDN PRI and SS7 in the *Outseize Control* message.

After configuring a channel with an Outseize Instruction List, the CSP uses the list to initiate an outseizure on a channel. You set this up by sending an *Outseize Control* message with a “Use Instruction List” ICB.

The last instruction in a list should always be “Wait For Host Control” or “Wait for Host Control with Answer Supervision,” where the CSP waits for further instructions from the host to continue processing.

NOTE: A seize instruction *cannot* be preprogrammed.

Related Messages

- *Outseize Control 0x002C* (OutseizeControl)
- *Call Control Instructions Query 0x0087* (CallControlInstructionQuery)

Sent by Host

Example Message (Socket Log Output for EXS SwitchKit)

The following socket log output/example message, shows the host sending an Outseize instruction list to the CSP to configure span 0x00, channels 0x00 - 0x17.

```
00 15 00 2A 00 00 ff 01 02 0d 03 00 00 00 0d 03 00 00 17
ff 00 00 00
```

Clear Instruction list

```
00 15 00 2A 00 00 ff 01 02 0d 03 00 00 00 0d 03 00 00 17
01 05 01 00
```

Outpulse stage 1

```
00 15 00 2A 00 00 ff 01 02 0d 03 00 00 00 0d 03 00 00 17
02 08 00 00
```

Send host acknowledgement

```
00 15 00 2A 00 00 ff 01 02 0d 03 00 00 00 0d 03 00 00 17
03 0f 00 00
```

Wait for host control w/Answer Supervision

SwitchKit Code Configuration

```
OutseizeInstrListConfig (  
    Node = integer,  
    Range = StartSpan:StartChan - EndSpan:EndChan,  
    InstrNum = integer,  
    InstrType = integer,  
    InstrData1 = integer,  
    InstrData2 = integer);
```

C Structure

```
typedef struct {  
    unsigned short StartSpan;  
    UBYTE StartChannel;  
    unsigned short EndSpan;  
    UBYTE EndChannel;  
    UBYTE InstrNum;  
    UBYTE InstrType;  
    UBYTE InstrData1;  
    UBYTE InstrData2;  
} XL_OutseizeInstrListConfig;
```

C++ Class

```
class XLC_OutseizeInstrListConfig : public  
    XLC_ChanRangeMessage {  
public:  
    unsigned short getStartSpan() const;  
    void setStartSpan(unsigned short x);  
    UBYTE getStartChannel() const;  
    void setStartChannel(UBYTE x);  
    unsigned short getEndSpan() const;  
    void setEndSpan(unsigned short x);  
    UBYTE getEndChannel() const;  
    void setEndChannel(UBYTE x);  
    UBYTE getInstrNum() const;  
    void setInstrNum(UBYTE x);  
    UBYTE getInstrType() const;  
    void setInstrType(UBYTE x);  
    UBYTE getInstrData1() const;  
    void setInstrData1(UBYTE x);  
    UBYTE getInstrData2() const;  
    void setInstrData2(UBYTE x);  
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x002A)	3, 4	Message Type (0x002A)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u>	8, 9	Status (MSB, LSB)
	Address Method	10	Checksum
	0x01 - Range of AEs		
	Number of AEs to follow		
AE			
	0x0D Channel (Starting)		
	0x0D Channel (Ending)		
:	Instruction Number You must assign each message an instruction number. The <i>Outseize Control</i> message specifies the first instruction in the list to be used and the CSP executes each message in order, by instruction number until the wait for host control instruction. Valid instruction numbers are 1 - 20. To clear all instructions from a channel, set the <i>Instruction Number</i> to 0xFF and <i>Instruction Type</i> to 0x00 (Null).		
:	Instruction Type 0x00 Null 0x01 Scan For Wink N 0x02 Scan for ANI Request Off-hook 0x03 Scan for Dialtone 0x04 Report Call Processing Event 0x05 Output Stage N Address Data 0x06 Wait for Host Address Data 0x07 Wait for Host Control Message 0x08 Send Host Acknowledgment 0x09 Do Call Progress Analysis 0x0D Cancel R2 Receiver 0x0E Scan for Backward R2 Signal 0x0F Wait for Host Control With Answer Supervision 0x10 Do CPA Without Line Signaling 0x11 Delay N Milliseconds		
:	Data[0] See Instruction Type Data Table below.		
:	Data[1] See Instruction Type Data Table below.		
:	Checksum		

Instruction Type Data

Instruction Type	Data 0	Data 1
0x00 Null	0x00	0x00
0x01 Scan for Wink N	Wink Number (0x01-0x08)	0x00
0x02 Scan for ANI Request Off-Hook	0x00	0x00
0x03 Scan for Dialtone	0x00	0x00
0x04 Report Call Processing Event	0x01 Off-hook 0x03 Wink 1 0x04 Wink 2 0x05 Wink 3 0x06 Wink 4 0x07 Wink 5 0x08 Wink 6 0x09 Wink 7 0x0A Wink 8 0x0B Dialtone Detected 0x0D Backward R2 Signal 0x0E First Digit Detected	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x05 Outpulse Stage N Address Data	0x01 Stage Number 1 0x02 Stage Number 2 0x03 Stage Number 3 0x04 Stage Number 4	0x00 0x00 0x00 0x00
0x06 Wait for Host Address Data	0x00	0x00
0x07 Wait for Host Control Message	0x00	0x00
0x08 Send Host Acknowledgment	0x00	0x00
0x09 Do Call Progress Analysis	0x00 North American Default 0x01 Dialtone 0x02 CPC Detection 0x03 Energy Detection	0x00 0x00 0x00 0x00
0x0D Cancel R2 Receiver	0x00	0x00
0x0E Scan for Backward R2 Signal	0x00	0x00
0x0F Wait for Host Control with Answer Supervision	0x00	0x00
0x10 Do CPA Without Line Signaling	0x00	0x00
0x11 Delay N Milliseconds	Delay Value, MSB: Values are in units of 10 ms. Max. value is 10 seconds (0x03E8).	Delay Value, LSB:

**Default Outseize
Instruction Lists**

The default instructions for T1 and E1 protocols are as follows:

T1

1. Wait For Host Control Message

E1

1. Outpulse Stage 1 Digits
2. Outpulse Stage 2 Digits
3. Outpulse Stage 3 Digits
4. Outpulse Stage 4 Digits
5. Report Call Processing Event
6. Wait For Host Control Message

PCM Encoding Format Configure 0x00D9

SwitchKit Name PCMEncodingConfig

Type EXS API and SwitchKit API message

Description **PCM Encoding Format Configure 0x00D9**

This message allows the host to set the PCM (pulse code modulation) encoding format of a channel to A-law or μ -law on T1 or E1 spans.

If an A-law encoded channel is connected to a μ -law encoded channel, the CSP performs the necessary conversion between the two channels to allow the appropriate voice data to be transmitted to both listeners.

Channels on E1 line cards default to A-law encoded format.

When you configure SS7 with an ANSI stack over an E1 Card, you must send the PCM Encoding Format Configure message with the PCM encoding format set to A-law **after** you send the SS7 Signaling Stack Configure (0x5C) message. The SS7 Signaling Stack Configure message overwrites the PCM Encoding Format Configure message.

Channels on T1 line cards default to μ -law encoded format.

When you configure SS7 with an ITU stack over a T1 Card or you configure Euro-ISDN over T-1, you must send the PCM Encoding Format Configure message with the PCM encoding format set to μ -law **after** you send the SS7 Signaling Stack Configure or the ISDN Interface Configure (0x60) messages. The ISDN Interface Configure message overwrites the PCM Encoding Format Configure message.

Occasionally, as in the case of a raw 8-bit data pipe connection, encoding conversion must be disabled. To do so, use this message to set both sides of the connection to the same PCM encoding format and no conversion will be performed.

Sent by Host

SwitchKit Code **Configuration**

```
PCMEncodingConfig (
    Node = integer,
    Range = StartSpan:StartChan - EndSpan:EndChan,
    Format = integer);
```

C Structure

```
typedef struct {
    unsigned short StartSpan;
    UBYTE StartChannel;
    unsigned short EndSpan;
    UBYTE EndChannel;
    UBYTE Format;
} XL_PCMEncodingConfig;
```

C++ Class

```
class XLC_PCMEncodingConfig : public XLC_ChanRangeMessage
{
public:
    unsigned short getStartSpan() const;
    void setStartSpan(unsigned short x);
    UBYTE getStartChannel() const;
    void setStartChannel(UBYTE x);
    unsigned short getEndSpan() const;
    void setEndSpan(unsigned short x);
    UBYTE getEndChannel() const;
    void setEndChannel(UBYTE x);
    UBYTE getFormat() const;
    void setFormat(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00D9)	3, 4	Message Type (0x00D9)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB	8, 9	Status MSB, LSB
	Address Method 0x00 - Individual Method	10	Checksum
	Number of AEs to follow		
	AEs 0x0D Channel (Starting) 0x0D Channel (Ending)		
:	Format 0x01 μ -law encoded PCM (North America & Japan) 0x02 A-law encoded PCM (Europe)		
:	Checksum		

Park Channel 0x00BF

SwitchKit Name ParkChannel

Type EXS API and SwitchKit API message

Description **Park Channel 0x00BF**

This message is used to park one or both channels involved in a connection. This message disconnects two parties but does not tear down the connection.

To park only one channel, enter the channel to be parked as both Channel A and Channel B in the AIB. When a channel is parked, it is not associated with any other channel and silence is transmitted to it. The channel may be used in a subsequent connection or released.

When parking two channels, the host will receive a positive acknowledgment to this message for the first channel and a *DSO Status Change* message with a channel status of “parked” for the second channel.

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short SpanA;
    UBYTE ChannelA;
    unsigned short SpanB;
    UBYTE ChannelB;
} XL_ParkChannel;
```

C++ Class

```
class XLC_ParkChannel : public XLC_OutboundMessage {
public:
    unsigned short getSpanA() const;
    void setSpanA(unsigned short x);
    UBYTE getChannelA() const;
    void setChannelA(UBYTE x);
    unsigned short getSpanB() const;
    void setSpanB(unsigned short x);
    UBYTE getChannelB() const;
    void setChannelB(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00BF)	3, 4	Message Type (0x00BF)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u> Address Method 0x00 - Individual AEs <hr/> Number of AEs 0x02 <hr/> AEs 0x0D Channel A 0x0D Channel B	8, 9	Status MSB, LSB
:	Checksum	10	Checksum

Play File Modify 0x011C

SwitchKit Name PlayFileModify

Type EXS API and SwitchKit API message

Description **Play File Modify 0x011C**

NOTE: This message applies to the DSP Series 2 card only.

Use this message to modify a current play file session. You can modify the gain and speed of the playback, pause or resume the playback, or skip forward or backward (in 100 ms increments) through the current file.

When modifying a file that is playing to a conference, use the Conference AIB instead of the Channel AIB. If you are modifying a file that is playing to a conference, use the optional File ID TLV. If you use the File ID TLV, all instances of that File ID being played in the specified conference are modified. If the File ID TLV is absent, all the files playing into that conference are modified.

Sent by Host Application

Related API Messages *Play File Start, Play File Stop*

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE Action;
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[220];
} XL_PlayFileModify;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[251];
} XL_PlayFileModifyAck;
```

C++ Class

```
class XLC_PlayFileModify : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getConferenceID() const;
```

```
void setConferenceID(XBYTE x);
XBYTE getParentConferenceID() const;
void setParentConferenceID(XBYTE x);
XBYTE getChildConferenceID() const;
void setChildConferenceID(XBYTE x);
XBYTE getSpan() const;
void setSpan(XBYTE x);
UBYTE getChannel() const;
void setChannel(UBYTE x);
UBYTE getAction() const;
void setAction(UBYTE x);
UBYTE getDataType() const;
void setData(UBYTE x);
UBYTE getTLVCount() const;
void setTLVCount(UBYTE x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x)
};
```

C++ Class Response

```
class XLC_PlayFileModifyAck : public XLC_OutboundMessage
{
public:

    unsigned short getStatus() const
    void setStatus(unsigned short x)
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    ;
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x011C)	3, 4	Message Type (0x011C)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB)
:	Address Method		0x0001 Invalid TLV Data Software can't find the TLV Data Buffer. This can also occur if the Data for a TLV is out of range. 0x0003 Invalid number of TLVs There are no TLVs in the message. 0x0004 Invalid TLV Length The TLV length is different from what is expected. 0x0006 Invalid TLV Unknown TLV 0x000D Mandatory TLVs missing One or more mandatory TLVs are missing Also see Common Response Status Values in the <i>API Reference</i>
	Number of AEs to follow	10	AIB (same as message)
	AEs	11	Checksum
	0x0D Channel		
	Or		
	0x55 Conference ID or 0x45 Child Conference ID and 0x01 Slot		
	If playing file to a conference		
:	Action		
	0x00 Modify		
	0x01 Pause		
	0x02 Resume		
	0x03 Skip Forward		
	0x04 Skip Backward		
:	Data Type		
	0x00 TLVs		

Play File Modify 0x011C

:	Number of TLVs to Follow
:	<p>TLVs</p> <p>If the Action field is Modify (0x00), these TLVs are optional:</p> <ul style="list-style-type: none">0x05E3 Gain0x05E4 Speed0x0604 DTMF Clamping/Filtering Enable <p>If the Action field is Pause (0x01) or Resume (0x02), no TLVs are used</p> <p>If the Action field is Skip Forward (0x03) or Skip Backward (0x04), this TLV is mandatory:</p> <ul style="list-style-type: none">0x05EC Skip Steps <p>If the AIB specifies Conference ID (0x55) or Child Conference ID (0x45) use this TLV to indicate the file to be modified:</p> <ul style="list-style-type: none">0x05E0 File ID
:	Checksum

Play File Start 0x011B

SwitchKit Name PlayFileStart

Type EXS API and SwitchKit API message

Description **Play File Start 0x011B**

NOTE: This message applies to the DSP Series 2 card only.

Use this message to start playing a file.

You can either specify the channel and the DSP Series 2 card, or let the CSP Matrix Series 3 Card choose the DSP card by leaving the DSP slot set to 0xFF. The CSP Matrix Series 3 Card reports back which DSP Series 2 card has been selected.

If you are accessing files from an external server, you must send the *Generic Card Configure* message (0x0122) to set the Vocabulary Index File before you send this message, because you must identify a file's ID and location before you can play it. Internal recordings are an exception to this rule. Internal recordings can be played without having their IDs in the Vocabulary Index File.

The *Play File Start* message may use File IDs from 0x00 to 0xFFFFFFFF. 0xFFFFFFFF is reserved by Dialogic for internal use, except when using media streaming over RTP, when the range is up to 0x0100000.

When recording a conference or playing a file to a conference, use the Conference AIB instead of the Channel AIB.

Sent by Host Application

Related API Messages *Play File Modify, Play File Stop*

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE FileCount;
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[220];
} XL_PlayFileStart;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved[13];
    UBYTE AddrInfo[251];
} XL_PlayFileStartAck;
```

C++ Class

```

class XLC_PlayFileStart : public XLC_OutboundMessage {
public:

    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    UBYTE getChannelSlot() const;
    void setChannelSlot(UBYTE x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    UBYTE getConferenceSlot() const;
    void setConferenceSlot(UBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getFileCount() const ;
    void setFileCount(UBYTE x) {;
    UBYTE getDataType() const ;
    void setDataType(UBYTE x) ;
    UBYTE getTLVCount() const ;
    void setTLVCount(UBYTE x) ;
    const UBYTE *getData() const ;
    UBYTE *getData() ;
    void setData(UBYTE *x) ;
};

```

C++ Class Response

```

class XLC_PlayFileStartAck : public XLC_OutboundMessage {
public:

    unsigned short getStatus() const
    void setStatus(unsigned short x)
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    UBYTE getChannelSlot() const;
    void setChannelSlot(UBYTE x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    UBYTE getConferenceSlot() const;
    void setConferenceSlot(UBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    ;

```

Hex API Message Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x011B)	3, 4	Message Type (0x011B)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs		
	Number of AEs to follow		
	AEs Use one of the following: 0x0D Channel 0x55 Conference ID 0x45 Child Conference ID Mandatory AE 0x01 Slot Note: If the slot is set to 0xFF, the CSP Matrix Series 3 Card chooses the slot.	8, 9	Status (MSB, LSB) 0x0001 Invalid TLV Data Software can't find the TLV Data Buffer. This can also occur if the Data for a TLV is out of range. 0x0003 Invalid number of TLVs There are no TLVs in the message. 0x0004 Invalid TLV Length The TLV length is different from what is expected. 0x0006 Invalid TLV Unknown TLV 0x000D Mandatory TLVs missing One or more mandatory TLVs are missing Also see Common Response Status Values in the <i>API Reference</i>
		10-14	AIB (same as message)
		15	Checksum
:	Number of files		
:	Data Type 0x00 TLVs		

Play File Start 0x011B

:	Number of TLVs to Follow
:	<p>TLVs</p> <p>Mandatory:</p> <p>For a single file:</p> <p style="padding-left: 20px;">0x05E0 File ID</p> <p>If the File ID is 0x00100000 or higher, you must also send:</p> <p style="padding-left: 20px;">0x05E1 File Format</p> <p style="padding-left: 20px;">0x05E2 File Location</p> <p style="padding-left: 20px;">0x05E4 Speed</p> <p>For multiple files:</p> <p>You must send one 0x05E0 File ID TLV for each file, and all File IDs must be lower than 0x00100000.</p> <p>Optional</p> <p style="padding-left: 20px;">0x05E3 Gain</p> <p style="padding-left: 20px;">0x05E4 Speed</p> <p style="padding-left: 20px;">0x05E5 Barge In</p> <p style="padding-left: 20px;">0x05E6 File Event Descriptor</p> <p style="padding-left: 20px;">0x05E7 Playback Repeat</p> <p style="padding-left: 20px;">0x05F2 Play File Queue</p> <p style="padding-left: 20px;">0x0604 DTMF Clamping/Filtering Enable</p> <p style="padding-left: 20px;">0x0614 Offset and Length</p> <p style="padding-left: 20px;">0x05F2 Play File Queue</p> <p><u>Media Streaming over RTP</u></p> <p>Mandatory TLVs</p> <p style="padding-left: 20px;">0x0687 - Enable RTP for Play/Record File</p> <p style="padding-left: 20px;">0x29FF Media Local End Point Information</p> <p style="padding-left: 20px;">0x2A00 Media Remote End Point Information</p> <p style="padding-left: 20px;">0x2A01 Media Per Stream Information</p> <p style="padding-left: 20px;">0x2A02 Media Per Codec Information</p> <p style="padding-left: 20px;">0x2A07 Media Port</p> <p style="padding-left: 20px;">0x2A0E Media Connection Address</p> <p>NOTE: if using RTP, the File ID must be below 0x00100000, so the File Format and File Location TLVs are not required.</p>
:	Checksum

Play File Stop 0x011D

SwitchKit Name PlayFileStop

Type EXS API and SwitchKit API message

Description **Play File Stop 0x011D**

NOTE: This message applies to the DSP Series 2 card only.

Use this message to end a current play file session. The message may be used to cancel only play file requests that were made with the new *Play File Start* message.

If you are stopping a play file to a conference (using the Conference ID AIB) you can use the File ID TLV. If File ID TLV is present, all instances of that file ID being played in the specified conference are stopped. If the File ID TLV is absent, all the files playing into that conference are stopped.

Sent by Host Application

Related API Messages *Play File Start, Play File Modify*

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_PlayFileStop;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[251];
} XL_PlayFileStopAck;
```

C++ Class

```
class XLC_PlayFileStop : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
};
```

```
XBYTE getSpan() const;
void setSpan(XBYTE x);
UBYTE getChannel() const;
void setChannel(UBYTE x);

UBYTE getDataType() const;
void setDataType(UBYTE x);
UBYTE getTLVCount() const;
void setTLVCount(UBYTE x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x)
};
```

C++ Class Response

```
class XLC_PlayFileStopAck : public XLC_OutboundMessage {
public:

    unsigned short getStatus() const
    void setStatus(unsigned short x)
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x011D)	3, 4	Message Type (0x011D)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB)
:	Address Method	10	AIB (same one returned as sent)
	0x00 - Individual AEs	11	Checksum
:	Number of AEs to follow		
:	AEs Use one of the following: 0x0D Channel 0x55 Conference ID (If playing file to a conference) 0x45 Child Conference ID (If playing file to child conference)		
:	Data Type 0x00 (2-byte Global TLVs)		
:	Number of TLVs		
:	TLVs If the AIB specifies Conference ID (0x55) or Child Conference ID (0x45), use this TLV to indicate the ID of the file to be stopped: 0x05E0 File ID		
:	Checksum		

Poll 0x00AB

SwitchKit Name	PollMessage
Type	EXS API and SwitchKit API message
Description	<p>Poll 0x00AB</p> <p>This message is sent by each CSP Matrix Series 3 Card to the host periodically, according to the poll interval.</p> <p>This message is also sent by each SS7 card (active and standby) to the SS7 local host periodically, according to the poll interval when interfaced by TCP/IP connections.</p> <p>The default interval for the CSP Matrix Series 3 Card is 2 seconds; it can be changed using the <i>Poll Interval Configure</i> message.</p> <p>The default interval for the SS7 card is 2 seconds; it can be changed using the <i>Poll Interval Configure</i> message.</p> <p>A <i>Poll</i> message is sent every time a state transition occurs. On transitions to the boot state, the poll may be delayed while the matrix or SS7 card resets. Every <i>Poll</i> message indicates the most recent matrix or SS7 card state transitions.</p> <p>The information in the <i>Poll</i> message is used by the host to detect that a matrix is attached to the host and that the communication link is functional. In a redundant system, information about the adjacent matrix or SS7 card state is also included. The host must use this information to properly communicate with the system.</p> <p>The ACK to this message can go to the standby CSP Matrix Series 3 Card.</p>
Sent by	CSP
SwitchKit Code	SwitchKit Code

C Structure

```
typedef struct {
    unsigned short Status;
    UBYTE SystemType;
    UBYTE MatrixSide;
    UBYTE MatrixState;
    UBYTE AdjMatrixState;
    UBYTE StatusBits;
    UBYTE Reserved1;
    UBYTE Reserved2;
    unsigned short ExcelPort;
} XL_PollMessage;
```

C++ Class

```
class XL_C_PollMessage : public XLC_InboundMessage {
public:
```

```

unsigned short getStatus() const;
void setStatus(unsigned short x);
UBYTE getSystemType() const;
void setSystemType(UBYTE x);
UBYTE getMatrixSide() const;
void setMatrixSide(UBYTE x);
UBYTE getMatrixState() const;
void setMatrixState(UBYTE x);
UBYTE getAdjMatrixState() const;
void setAdjMatrixState(UBYTE x);
UBYTE getStatusBits() const;
void setStatusBits(UBYTE x);
UBYTE getReserved1() const;
void setReserved1(UBYTE x);
UBYTE getReserved2() const;
void setReserved2(UBYTE x);
unsigned short getExcelPort() const;
void setExcelPort(unsigned short x);
};

```

Overview of message

The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0005)
3, 4	Message Type (0x00AB)	3, 4	Message Type (0x00AB)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8, 9	Status MSB, LSB	8	Checksum
10	Poll Source		
11	Data		
:	:		
12	Checksum		

EXS API Hex Format - Detailed

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0005)
3, 4	Message Type (0x00AB)	3, 4	Message Type (0x00AB)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8, 9	Status MSB, LSB The status is always 0x0010	8	Checksum
10	System Type 0x01 CSP 2090 0x03 CSP 2040 0x04 - EXNET Connect 0x80 SS7 card (both PQ and Series 3)		
11	Data The data varies, depending on the Poll source. The following data is reported if the Poll source is an CSP Matrix Series 3 Card, EXNET-ONE card, or EXNET Connect® card. Data[0] Card ID This field indicates the CSP Matrix Series 3 Card that is polling. 0x00 Right CSP 2000 Matrix (Slot 0x20) 0x01 Left CSP 2000 Matrix (Slot 0x21) Data[1] Card State This field indicates the most recent matrix state. 0x01 Booting 0x02 Initializing 0x03 Standby 0x04 Switching Over 0x05 Active		

Data[2] Adjacent Card State

This field indicates the most recent adjacent CSP Matrix Series 3 Card state change.

Changes in the adjacent CSP Matrix Series 3 Card state should be considered as events at the host.

- 0x00 No adjacent CSP Matrix Series 3 Card detected.
- 0x01 Adjacent CSP Matrix Series 3 Card detected, cannot establish inter-matrix communications.
- 0x02 Adjacent CSP Matrix Series 3 Card detected, inter-matrix communications established, and adjacent CSP Matrix Series 3 Card cannot establish communications with the host.
- 0x03 Adjacent CSP Matrix Series 3 Card detected, inter-matrix communications established, and adjacent CSP Matrix Series 3 Card communicating with host.

Data[3] Card Status

This field is a bit mask. The meaning of each bit is as follows:

- Bit 0 Host Messaging
 - 0 Receiving messages from host (at least 1 per 5-second interval)
 - 1 Not Receiving messages from the host
- Bit 1* Download
 - 0 Download not required
 - 1 Download required
- Bit 2 Active Ethernet Interface
 - 0 Interface 1 Active
 - 1 Interface 2 Active
- Bit 3 Configuration
 - 0 Ready for configuration messages
 - 1 Not ready for configuration messages
- Bit 4 Reserved
- Bit 5 TFTP Download
 - 0 Not In Progress
 - 1 In Progress
- Bits 6-7 Reserved

* If the CSP Matrix Series 3 Card has system software, or if it can obtain it from the adjacent CSP Matrix Series 3 Card, Bit 1 is "0". If Bit 1 is set, the host initiates a download of the system software. In some circumstances, both CSP Matrix Series 3 Card cards indicate that they require system software to be downloaded. The CSP Matrix Series 3 Card that receives its *Download Complete* message last will become the active CSP Matrix Series 3 Card.

Data[4, 5] Reserved

Data[6, 7] Multi-Host Connection Status

This field is a bit mask indicating the connection status of each host according to the previous *Poll* message sent by the CSP. If the *Poll* message was acknowledged by a specific host port, the corresponding bit is set to 1. If the *Poll* message was not acknowledged by a specific host, the corresponding bit is set to 0.

NOTE: Polling is operational only when the host is connected to Host Port 3142 and polling is enabled.

Status

- 0 - no *Poll* ACK received
- 1 - *Poll* ACK received

<u>Bit Number</u>	<u>Host Port Number</u>
Byte 1	
1 (Low Bit)	0x3142
2	0x3143
3	0x3144
4	0x3145
5	0x3146
6	0x3147
7, 8	Reserved (0)
Byte 2	
9-16	Reserved (0)

<p><u>The following data is reported if the Poll Source is an SS7 card:</u></p>	
	Data[0] Slot Number
	Data[1] Card State
	0x01 Booting or Unknown
	0x02 Synchronizing
	0x03 Standby
	0x04 Reserved
	0x05 Active or Single State
	Data[2] Adjacent Card State
	0x01 Booting or Unknown
	0x02 Synchronizing
	0x03 Standby
	0x04 Reserved
	0x05 Active or Single State
	Data[3] Card Status
	This field is a bit mask. The meaning of each bit is as follows:
	<u>Bit</u>
	Bit 0 Host Messaging
	0 Receiving messages from local host (default)
	1 Not Receiving messages from the host
	Bit 1 Reserved
	Bit 2 Standby Card Messaging
	0 Mate SS7 card receiving message from local host (default)
	1 Mate SS7 card not receiving message from local host
	3 Configuration
	0 - Ready for configuration messages
	1 - Not ready for configuration messages
	4-7 Reserved
	Data[4,5] Reserved
12	Checksum

Poll Interval Configure 0x009F

SwitchKit Name	PollIntervalConfig
Type	EXS API and SwitchKit API message
Description	<p>Poll Interval Configure 0x009F</p> <p>The poll interval is the time between <i>Poll</i> messages generated by the CSP Matrix Series 3 Card or other cards. A CSP Matrix Series 3 Card may generate <i>Poll</i> messages at any time to report CSP Matrix Series 3 Card state changes. The default poll interval is two seconds. This message can also be sent from the host to an SS7 card to change the SS7 poll interval.</p> <p>The host can send this message to the active or standby CSP Matrix Series 3 Card.</p>
Sent by	Host
SwitchKit Code	<p>Configuration</p> <pre>PollIntervalConfig (Node = integer, PollInterval = integer);</pre> <p>C Structure</p> <pre>typedef struct { UBYTE PollInterval; } XL_PollIntervalConfig;</pre> <p>C++ Class</p> <pre>class XLC_PollIntervalConfig : public XLC_OutboundMessage { public: UBYTE getPollInterval() const; void setPollInterval(UBYTE x); };</pre>

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0006)
3, 4	Message Type (0x009F)	3, 4	Message Type (0x009F)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status MSB, LSB
		10	Checksum
8	Poll Interval 0x00 Disable Polling (except when a CSP Matrix Series 3 Card state transition occurs) If Host Link Failure Detection is enabled, you must disable it using the <i>System Configuration</i> message before polling is actually disabled. 0x01–0xFF The number of seconds for the polling interval, in tenths of seconds (the maximum polling interval is 25.5 seconds). The poll interval remains set after a reset, but must be reset if the CSP Matrix Series 3 Card powered down.		
9	Checksum		

Poll Request 0x009E

SwitchKit Name PollRequest

Type EXS API and SwitchKit API message

Description **Poll Request 0x009E**

Use this message to initiate the sending of a *Poll* message from the CSP Matrix Series 3 Card or SS7 card to the host. The host may use this message to implement its own polling scheme, or to obtain information from the CSP Matrix Series 3 Card or SS7 card.

The host can send this message to the active or standby CSP Matrix Series 3 Card.

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    } XL_PollRequest;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE SystemType;
    UBYTE MatrixSide;
    UBYTE MatrixState;
    UBYTE AdjMatrixState;
    UBYTE StatusBits;
    UBYTE Reserved1;
    UBYTE Reserved2;
    unsigned short ExcelPort;
    } XL_PollRequestAck;
```

C++ Class

```
class XLC_PollRequest : public XLC_OutboundMessage {
public:
    };
```

C++ Class Response

```
class XLC_PollRequestAck : public XLC_AcknowledgeMessage
{
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getSystemType() const;
    void setSystemType(UBYTE x);
    UBYTE getMatrixSide() const;
    void setMatrixSide(UBYTE x);
```

```

UBYTE getMatrixState() const;
void setMatrixState(UBYTE x);
UBYTE getAdjMatrixState() const;
void setAdjMatrixState(UBYTE x);
UBYTE getStatusBits() const;
void setStatusBits(UBYTE x);
UBYTE getReserved1() const;
void setReserved1(UBYTE x);
UBYTE getReserved2() const;
void setReserved2(UBYTE x);
unsigned short getExcelPort() const;
void setExcelPort(unsigned short x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x000E)
3, 4	Message Type (0x009E)	3, 4	Message Type (0x009E)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Checksum	8, 9	Status MSB, LSB (Response continued below.)
10	System Type 0x01 CSP 2000 0x03 CSP 1000		
11	CSP Matrix Series 3 Card ID This field indicates the CSP Matrix Series 3 Card that is polling. 0x00 Right CSP Matrix Series 3 Card (Slot 0x20) 0x01 Left CSP Matrix Series 3 Card (Slot 0x21)		
12	CSP Matrix Series 3 Card State This field indicates the most recent matrix state. 0x01 Booting 0x02 Initializing 0x03 Standby 0x04 Switching Over 0x05 Active		

13	<p>Adjacent CSP Matrix Series 3 Card State</p> <p>This field indicates the most recent adjacent CSP Matrix Series 3 Card state change. Changes in the adjacent CSP Matrix Series 3 Card state should be considered as events at the host.</p> <p>0x00 No adjacent CSP Matrix Series 3 Card detected.</p> <p>0x01 Adjacent CSP Matrix Series 3 Card detected, cannot establish inter-matrix communications.</p> <p>0x02 Adjacent CSP Matrix Series 3 Card detected, inter-matrix communications established, and adjacent CSP Matrix Series 3 Card cannot establish communications with the host.</p> <p>0x03 Adjacent CSP Matrix Series 3 Card detected, inter-matrix communications established, and adjacent CSP Matrix Series 3 Card communicating with host.</p>
14	<p>Status Bits</p> <p>This field is a bit mask. The meaning of each bit is as follows:</p> <p>Bit 0 Host Messaging 0 Receiving messages from host (at least 1 per 5-second interval) 1 Not Receiving messages from the host</p> <p>Bit 1* Download 0 Download not required 1 Download required</p> <p>Bit 2 Reserved</p> <p>Bit 3 Configuration 0 Ready for configuration messages 1 Not ready for configuration messages</p> <p>Bit 4 Power Down 0 Ready for power down 1 Not ready for power down</p> <p>Bit 5 TFTP Download 0 Not In Progress 1 In Progress</p> <p>Bits 6-7 Reserved</p> <p>* If the matrix has system software, or if it can obtain it from the adjacent matrix, Bit 1 is "0". If Bit 1 is set, the host initiates a download of the system software. In some circumstances, both CSP Matrix Series 3 Cards indicate that they require system software to be downloaded. The CSP Matrix Series 3 Card that receives its <i>Download Complete</i> message last will become the active CSP Matrix Series 3 Card.</p>
15, 16	Reserved
17	Checksum

PPL Assign 0x00D1

SwitchKit Name	PPLAssign
Type	EXS API and SwitchKit API message
Description	<p>PPL Assign 0x00D1</p> <p>This message assigns a custom protocol to a channel. Before you can assign a custom protocol to a channel, you must:</p> <ul style="list-style-type: none"> • Create and download all tables used by the custom protocol. • Take the channel out-of-service. <p>When a new load of system software is downloaded to the CSP, you must re-download and reassign protocols.</p>

Sent by Host

SwitchKit Code Configuration

```
PPLAssign (
    Node = integer,
    StartSpan = integer,
    StartChannel = integer,
    EndSpan = integer,
    Range = StartSpan:StartChan - EndSpan:EndChan,
    EndChannel = integer,
    Span = integer,
    Channel = integer,
    Subrate = integer,
    StackID = integer,
    LinkID = integer,
    Destination = integer,
    Route = integer,
    SSLGroupID = integer,
    SSLProfileID = integer,
    SSLServiceNumber = integer,
    V5ID = integer,
    ComponentID = integer,
    ProtocolID = integer);
```

C Structure

```
typedef struct {
    UBYTE AddrInfo[30];
    unsigned short ComponentID;
    UBYTE ProtocolID;
} XL_PPLAssign;
```

C++ Class

```
class XLC_PPLAssign : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
```

```
void setAddrInfo(UBYTE *x);
XBYTE getStartSpan() const;
void setStartSpan(XBYTE x);
UBYTE getStartChannel() const;
void setStartChannel(UBYTE x);
XBYTE getEndSpan() const;
void setEndSpan(XBYTE x);
UBYTE getEndChannel() const;
void setEndChannel(UBYTE x);
XBYTE getSpan() const;
void setSpan(XBYTE x);
UBYTE getChannel() const;
void setChannel(UBYTE x);
XBYTE getV5ID() const;
void setV5ID(XBYTE x);
unsigned short getComponentID() const;
void setComponentID(unsigned short x);
UBYTE getProtocolID() const;
void setProtocolID(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)																															
Byte	Field Description	Byte	Field Description																														
0	Frame (0xFE)	0	Frame (0xFE)																														
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)																														
3, 4	Message Type (0x00D1)	3, 4	Message Type (0x00D1)																														
5	Reserved (0x00)	5	Reserved (0x00)																														
6	Sequence Number	6	Same Sequence Number																														
7	Logical Node ID	7	Logical Node ID																														
:	<u>AIB</u> Address Method Depends on component being addressed.																																
	Number of AEs to follow																																
	AE The AE used depends on the component being addressed. See PPL Component Addressing in the <i>API Reference</i> .	8, 9	Status MSB, LSB																														
		10	Checksum																														
:	PPL Component ID MSB, LSB See PPL Component IDs in the <i>API Reference</i> .																																
:	PPL Protocol ID Protocol IDs 1 - 10 are reserved per component for custom protocols. Shown below are two components and their default protocols that are included on the CSP. Default protocols are assigned through other methods such as <i>Trunk Type Configure</i> for the T1 protocols.																																
	<table border="1"> <thead> <tr> <th><u>Component ID</u></th> <th><u>Protocol ID</u></th> <th></th> </tr> </thead> <tbody> <tr> <td>Any</td> <td>0x01–0x0A</td> <td>Host-defined protocols</td> </tr> <tr> <td>0x01 E1</td> <td>0x0C*</td> <td>ITU-T Compelled R2</td> </tr> <tr> <td></td> <td>0x0D</td> <td>Clear</td> </tr> <tr> <td>0x03 T1</td> <td>0x0C*</td> <td>E&M</td> </tr> <tr> <td></td> <td>0x0D</td> <td>FXS Loop Start</td> </tr> <tr> <td></td> <td>0x0E</td> <td>FXS Ground Start</td> </tr> <tr> <td></td> <td>0x0F</td> <td>FXO Loop Start</td> </tr> <tr> <td></td> <td>0x10</td> <td>FXO Ground Start</td> </tr> <tr> <td></td> <td>0x11</td> <td>Clear</td> </tr> </tbody> </table> <p>* default</p>			<u>Component ID</u>	<u>Protocol ID</u>		Any	0x01–0x0A	Host-defined protocols	0x01 E1	0x0C*	ITU-T Compelled R2		0x0D	Clear	0x03 T1	0x0C*	E&M		0x0D	FXS Loop Start		0x0E	FXS Ground Start		0x0F	FXO Loop Start		0x10	FXO Ground Start		0x11	Clear
<u>Component ID</u>	<u>Protocol ID</u>																																
Any	0x01–0x0A	Host-defined protocols																															
0x01 E1	0x0C*	ITU-T Compelled R2																															
	0x0D	Clear																															
0x03 T1	0x0C*	E&M																															
	0x0D	FXS Loop Start																															
	0x0E	FXS Ground Start																															
	0x0F	FXO Loop Start																															
	0x10	FXO Ground Start																															
	0x11	Clear																															
:	Checksum																																

PPL Audit Configure 0x00DC

SwitchKit Name	PPLAuditConfig
Type	EXS API and SwitchKit API message
Description	<p>PPL Audit Configure 0x00DC</p> <p>This message configures the PPL auditing features on a card. Use the <i>PPL Audit Query</i> host message to retrieve the audit log.</p>
Sent by	Host

SwitchKit Code Configuration

```
PPLAuditConfig (
    Node = integer,
    Slot = integer,
    V5ID = integer,
    ComponentID = integer,
    AuditType = integer,
    Span = integer,
    Channel = integer);
```

C Structure

```
typedef struct {
    UBYTE AddrInfo[30];
    unsigned short ComponentID;
    UBYTE AuditType;
} XL_PPLAuditConfig;
```

C++ Class

```
class XLC_PPLAuditConfig : public XLC_SlotMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    XBYTE getV5ID() const;
    void setV5ID(XBYTE x);
    unsigned short getComponentID() const;
    void setComponentID(unsigned short x);
    UBYTE getAuditType() const;
    void setAuditType(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00DC)	3, 4	Message Type (0x00DC)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs 0x01 - Slot Number of AEs to follow AE 0x0D Channel	8, 9	Status MSB, LSB
:	PPL Component ID MSB, LSB See PPL Component IDs in the <i>API Reference</i> .	10	Checksum
:	<p>Audit Type This field is a bit mask that enables audit features. The bit values are 0=Disabled, 1=Enabled. (To enable a feature, set the bit to 1).</p> <p><u>Bit</u></p> <p>0 PPL Auditing Log</p> <p>1 PPL Error Alarm Use this bit to inform the host of a PPL state machine error through an Alarm message. If you enable this bit, you must also enable the PPL Auditing Log bit (Bit 0 above).</p> <p>2 PPL Individual Auditing Enable Use this bit to enable and disable individual PPL auditing, per component on the card containing the AIB entity.</p> <p>3 PPL Individual Entity Auditing Configure Use this bit to enable and disable auditing for the specific entity of the component.</p> <p>4-7 Reserved, must be 0</p> <p>Note: Bit 3 used in conjunction with Bit 2 enables auditing for the specific entity.</p> <p>Audit Type Validation: 0x00, 0x01, 0x02, 0x04: Valid with Slot or Entity AIB 0x03: Valid with Slot AIB only 0x06, 0x08, 0x0A, 0x0C, 0x0E: Valid with Entity AIB only 0x05, 0x07, 0x09, 0x0B, 0x0D, 0x0F - 0xFF: Not valid</p> <p>Note: An invalid Audit Type will return a NACK (0x17)</p>		
:	Checksum		

PPL Audit Query 0x00DD

SwitchKit Name PPLAuditQuery

Type EXS API and SwitchKit API message

Description **PPL Audit Query 0x00DD**

The host uses this message to request an audit for individual PPL components or entities. The CSP maintains two logs: the PPL state transition log and the PPL error log. Therefore, the host can use this message for two types of audits:

1. PPL State Transition audit
This audit retrieves information on a per-instantiated PPL component basis.
2. PPL Error audit
This audit retrieves information on a per-card basis.

The response to this message is the PPL audit. Its contents, including the length of the audit entry, varies per PPL component.

PPL component audits consist of one or more audit blocks, which in turn are made up of one or more audit entries. Each one of these messages queries one audit block. In order to audit a PPL component completely, the host must send several of these messages.

When there are no audit entries remaining, the response status to the *PPL Audit Query* message will indicate 0x00D7 (End of PPL Audit Data).

Prior to using this message, you must enable auditing using the *PPL Audit Configure* message.

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    unsigned short ComponentID;
    UBYTE AuditType;
    UBYTE AuditBlock;
} XL_PPLAuditQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE AddrInfo[30];
    unsigned short ComponentID;
    UBYTE AuditType;
    UBYTE AuditBlock;
    UBYTE AuditEntryLength;
    UBYTE NumberAuditEntries;
```

```

    UBYTE AuditEntries[215];
} XL_PPLAuditQueryAck;

```

C++ Class

```

class XL_C_PPLAuditQuery : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    XBYTE getV5ID() const;
    void setV5ID(XBYTE x);
    unsigned short getComponentID() const;
    void setComponentID(unsigned short x);
    UBYTE getAuditType() const;
    void setAuditType(UBYTE x);
    UBYTE getAuditBlock() const;
    void setAuditBlock(UBYTE x);
};

```

C++ Class Response

```

class XL_C_PPLAuditQueryAck : public XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    XBYTE getV5ID() const;
    void setV5ID(XBYTE x);
    unsigned short getComponentID() const;
    void setComponentID(unsigned short x);
    UBYTE getAuditType() const;
    void setAuditType(UBYTE x);
    UBYTE getAuditBlock() const;
    void setAuditBlock(UBYTE x);
    UBYTE getAuditEntryLength() const;
    void setAuditEntryLength(UBYTE x);
    UBYTE getNumberAuditEntries() const;
    void setNumberAuditEntries(UBYTE x);
    const UBYTE *getAuditEntries() const;
    UBYTE *getAuditEntries();
    void setAuditEntries(UBYTE *x);
};

```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00DD)	3, 4	Message Type (0x00DD)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB (starting with byte 0)	8, 9	Status MSB, LSB
:	:	10	AIB (starting with byte 0)
:	PPL Component ID MSB, LSB	:	PPL Component ID (MSB, LSB)
:	Audit Type	:	Audit Type
:	Audit Block Number	:	Audit Block Number
:	Checksum	:	Number of Audit Entries in this response.
		:	Audit Entry Length
		:	Audit Entry 0: Data[0]
		:	:
		:	Audit Entry 1: Data[0]
		:	:
		:	Audit Entry (n-1): Data[0]
		:	:
		N+3	Checksum

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00DD)	3, 4	Message Type (0x00DD)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID

:	<u>AIB</u> Address Method 0x00 - Individual AEs 0x01 - Slot <hr/> Number of AEs to follow <hr/> AE 0x0D Channel	8, 9	Status MSB, LSB
:	PPL Component ID MSB, LSB See PPL Component IDs in the <i>API Reference</i> .		
:	Audit Type 0x00 PPL State Transition Audit 0x01 PPL Error Audit 0x02 Audit Configuration	10	AIB 0x0D Channel The response can contain the AIB in two areas: -Immediately following the <i>Status</i> field: The address type that immediately follows the <i>Status</i> field is the same as the address type that the host included in the message. -Within each audit entry: The response may use any address type within each audit entry.
:	Audit Block Number The audit block you are querying. Begin an audit by specifying Audit Block Number 0 to retrieve the entire log. Repeatedly send this message, incrementing the block number, until the response status is 0x00D7 (PPL End of Auditing Data).	:	PPL Component ID (MSB, LSB)
:	Checksum	:	Audit Type 0x00 PPL State Transition Audit 0x01 PPL Error Audit 0x02 Audit Configuration The 0x02 audit type is a bit mask that enables audit configuration settings. The bit values are 0=Disabled, 1=Enabled. (To enable a feature, set the bit to 1). <u>Bit</u> 0 PPL Auditing Log 1 PPL Error Alarm 2 PPL Individual Log 3 PPL Entity Audit Log* 4 PPL Entity Error Alarm* * Bits 3 and 4 are invalid in Slot AIB.
		:	Audit Block Number
Response continued below.			
:	Number of Audit Entries in this response. You can calculate the number of data bytes that follow this field by multiplying the values of this field and the Audit Entry Length field.(n)		

:	<p>Audit Entry Length The number of bytes in each audit entry.</p>
:	<p>Audit Entry 0: Data[0] The audit data varies by audit type and PPL Component. The data is returned for each audit entry in the block. See the text below this table for specific data listed by Audit Type/Entry.</p> <p>If there is no audit information for an entry, all data field values are 0x00. If the value of the <i>Audit Type</i> field is 0x00, valid entries for the <i>Data</i> fields are as follows:</p> <p>E1/T1 PPL Components</p> <ul style="list-style-type: none"> Data[0] PPL Protocol ID Data[1] State Status Data[2, 3] PPL Event (MSB, LSB) Data[4] Initial State Data[5] Next State Data[6, 7] Timestamp (MSB, LSB) Data[8, 9] PPL Error Status (MSB, LSB) <p>ISDN/DASS 2 PPL Components</p> <ul style="list-style-type: none"> Data[0] Subrate Data[1, 2] Call Reference (MSB, LSB) Data[3] SAPI Data[4] CES Data[5] PPL Protocol ID Data[6] State Status Data[7, 8] PPL Event (MSB, LSB) Data[9] Initial State Data[10] Next State Data[11, 12] Timestamp (MSB, LSB) Data[13, 14] PPL Error Status (MSB, LSB) <p>SS7 Components</p> <ul style="list-style-type: none"> Data[0] PPL Protocol ID Data[1] State Status Data[2, 3] PPL Event (MSB, LSB) Data[4] Initial State (If State value is greater than 255, this is truncated to 0xFF) Data[6] Next State (If State value is greater than 255, this is truncated to 0xFF) Data[6, 7] Timestamp (MSB, LSB) Data[8, 9] PPL Error Status (MSB, LSB)

For error audits, AIBs are retrieved for each audit entry.
If the value of the *Audit Type* field is 0x01, valid entries for the *Data* fields are as follows:

E1/T1 PPL Components

Data[0]	AIB
:	:
Data[:]	PPL Protocol ID
Data[:]	State Status
Data[:]	PPL Event (MSB, LSB)
Data[:]	Initial State
Data[:]	Next State
Data[:]	Timestamp (MSB, LSB)
Data[:]	PPL Error Status (MSB, LSB)

ISDN/DASS 2 PPL Components

Data[0]	AIB
:	:
Data[:]	Reserved
Data[:]	Call Reference (MSB, LSB)
Data[:]	Reserved
Data[:]	Reserved
Data[:]	PPL Protocol ID
Data[:]	State Status
Data[:]	PPL Event (MSB, LSB)
Data[:]	Initial State
Data[:]	Next State
Data[:]	Timestamp (MSB, LSB)
Data[:]	PPL Error Status (MSB, LSB)

SS7 Components

The data begins with an AIB:

Data[0]	AIB
:	:
Data[:]	PPL Protocol ID
Data[:]	State Status
Data[:]	PPL Event (MSB, LSB)
Data[:]	Initial State
Data[:]	Next State
Data[:]	Timestamp (MSB, LSB)
Data[:]	PPL Error Status (MSB, LSB)

If the value of the *Audit Type* field is **0x02**, valid entries for the *Data* fields are as follows:

E1/T1 PPL Components

Data[0] PPL Protocol ID
Data[1] State Status
Data[2, 3] PPL Event (MSB, LSB)
Data[4] Initial State (If State value is greater than 255, this is truncated to 0xFF)
Data[5] Next State (If State value is greater than 255, this is truncated to 0xFF)
Data[6, 7] Timestamp (MSB, LSB)
Data[8, 9] PPL Error Status (MSB, LSB)
Data[10, 11] Extended Initial State (MSB, LSB)
Data[12, 13] Extended Next State (MSB, LSB)

ISDN/DASS 2 PPL Components

Data[0] Subrate
Data[1, 2] Call Reference (MSB, LSB)
Data[3] SAPI
Data[4] CES
Data[5] PPL Protocol ID
Data[6] State Status
Data[7, 8] PPL Event (MSB, LSB)
Data[9] Initial State (If State value is greater than 255, this is truncated to 0xFF)
Data[10] Next State (If State value is greater than 255, this is truncated to 0xFF)
Data[11, 12] Timestamp (MSB, LSB)
Data[13, 14] PPL Error Status (MSB, LSB)
Data[15, 16] Extended Initial State (MSB, LSB)
Data[17, 18] Extended Next State (MSB, LSB)

SS7 Components

Data[0] PPL Protocol ID
Data[1] State Status
Data[2, 3] PPL Event (MSB, LSB)
Data[4] Initial State (If State value is greater than 255, this is truncated to 0xFF)
Data[6] Next State (If State value is greater than 255, this is truncated to 0xFF)
Data[6, 7] Timestamp (MSB, LSB)
Data[8, 9] PPL Error Status (MSB, LSB)
Data[10, 11] Extended Initial State (MSB, LSB)
Data[12, 13] Extended Next State (MSB, LSB)

If the value of the *Audit Type* field is 0x03, valid entries for the *Data* fields are as follows:

E1/T1 PPL Components

Data[0]	AIB
:	:
Data[:]	PPL Protocol ID
Data[:]	State Status
Data[:]	PPL Event (MSB, LSB)
Data[:]	Initial State (If State value is greater than 255, this is truncated to 0xFF)
Data[:]	Next State (If State value is greater than 255, this is truncated to 0xFF)
Data[:]	Timestamp (MSB, LSB)
Data[:]	PPL Error Status (MSB, LSB)
Data[:]	Extended Initial State (MSB, LSB)
Data[:]	Extended Next State (MSB, LSB)

ISDN/DASS 2 PPL Components

Data[0]	AIB
:	:
Data[:]	Reserved
Data[:]	Call Reference (MSB, LSB)
Data[:]	Reserved
Data[:]	Reserved
Data[:]	PPL Protocol ID
Data[:]	State Status
Data[:]	PPL Event (MSB, LSB)
Data[:]	Initial State (If State value is greater than 255, this is truncated to 0xFF)
Data[:]	Next State (If State value is greater than 255, this is truncated to 0xFF)
Data[:]	Timestamp (MSB, LSB)
Data[:]	PPL Error Status (MSB, LSB)
Data[:]	Extended Initial State (MSB, LSB)
Data[:]	Extended Next State (MSB, LSB)

SS7 Components

The data begins with an AIB:

Data[0]	AIB
:	:
Data[:]	PPL Protocol ID
Data[:]	State Status
Data[:]	PPL Event (MSB, LSB)
Data[:]	Initial State (If State value is greater than 255, this is truncated to 0xFF)
Data[:]	Next State (If State value is greater than 255, this is truncated to 0xFF)
Data[:]	Timestamp (MSB, LSB)
Data[:]	PPL Error Status (MSB, LSB)
Data[:]	Extended Initial State (MSB, LSB)
Data[:]	Extended Next State (MSB, LSB)

	VDAC-ONE and IP Network Interface Series 2 Components
	Data[0] AIB
	: :
	Data[:] PPL Protocol ID
	Data[:] Blocked
	Data[:] PPL Event (MSB, LSB)
	Data[:] Initial State
	Data[:] Next State
	Data[:] Timestamp (MSB, LSB)
	Data[:] PPL Error Status (MSB, LSB)
	Data[:] Extended Initial State (MSB, LSB)
	Data[:] Extended Next State (MSB, LSB)
:	Audit Entry 1: Data[0]
:	:
:	Checksum

State Status

This byte indicates whether or not a blocking atomic function has been invoked by the state machine. A blocking atomic function is used for allocation of any off-board service resource required by the PPL. Once the result of the resource request is received, the channel unblocks and resumes the remainder of the primitive. Blocking is performed on a channel basis without affecting other channels.

- 0x00 Normal
- 0xFF Blocking

Timestamp

The timestamp is an incrementing 10 ms timer representing the relative time between PPL events. After reaching the value of 0xFFFF, the timer resets to zero.

PPL Error Status

Valid entries for this word-length field are as follows:

- 0x0000 No Error
- 0xF448 Invalid event for next state
- 0xF447 Invalid event for initial state
- 0xF446 Blocked message
- 0xF445 Blocked event
- 0xF444 Invalid event for blocked state
- 0xF443 Internal error (wrong task ID)
- 0xF442 Internal error (invalid header)
- 0xF441 Internal error (invalid header)
- 0xF440 Invalid event
- 0xF43E Timer is disabled
- 0xF43D Timer expired out of sequence
- 0xF43C Event not processed
- 0xF43B Internal error (invalid message)

- 0xF43A Cannot create new context
- 0xF439 Cannot enter new context
- 0xF438 Entering new context
- 0xF437 Leaving new context
- 0xF436 Invalid context
- 0xF435 No state/event list
- 0xF434 State machine suspended
- 0xF433 Timer expired while in blocked state
- 0xF432 Invalid response to blocked state
- 0xF431 Not waiting on internal event
- 0xF430 Initiated Channel purge
- 0xF42F Maximum iterations of state machine exceeded
- 0xF42E Null primitive
- 0xF42D Context is inconsistent
- 0xF42C Maximum nesting level exceeded
- 0xF42B Initiated state machine reset
- 0xF42A Maximum config byte offset exceeded
- 0xF429 Maximum number of GPRs exceeded
- 0xF420 Block Invalid Events
- 0xF41F Invalid Event Handler Activated

Example Message

This example shows a *PPL Audit Query* message that is being used to query a PPL State Transition audit log on logical span 0/channel 0, ISDN component #5. The message from the host to the CSP is as follows:

Header:

fe [00 16] [00 dd] 00 00 ff

AIB:

00 01 0D 03 00 00 00

Info:

[00 05] 00 00

MESSAGE		
Byte	Field Description	Value and Indication
0	Frame	0xFE
1	Length (MSB, LSB)	0x0016 (22)
2		
3	Message Type (MSB, LSB)	0x00DD (PPL Audit Query)
4		
5	Reserved	0x00
6	Sequence #	0x00

7	Logical Node ID	0xFF
8	AIB	0x00 (Single Entity)
9	Number of Address Elements	0x01
10	Address Type	0x0D (Channel)
11	Data Length	0x03
12	Logical Span (MSB, LSB)	0x0000 (Span 0)
13		
14	Channel	0x00 (Channel 0)
15	PPL Component ID (MSB, LSB)	0x0005 (ISDN L3P Call Reference)
16		
17	Audit Type	0x00 (PPL State Transition Audit)
18	Audit Block	0x00 (Block 0)
19	Checksum	0xC5 (does not appear in trace)

The trace of the response shows that 14 bytes of data are returned for each of nine audits on the specified channel/PPL component. The response, containing the log information, is as follows:

Header:

fe [00 92] [00 dd] 00 15 ff [00 10]

AIB:

00 01 0d 03 00 00 00

Info:

[00 05] 00 00 0f 09

Audit Entry 0:

00 00 00 00 0b 00 00 56 00 16 1d cd 00 00

Audit Entry 1:

00 00 00 00 0b 00 00 c9 16 06 1d cd 00 00

Audit Entry 2:

00 00 00 01 0b 00 00 67 06 1a 1e 96 00 00

Audit Entry 3:

00 00 00 01 0b 00 00 c8 1a 0b 1e 96 00 00

Audit Entry 4:

00 00 00 01 0b 00 00 c8 0b 06 1e 96 00 00

Audit Entry 5:

00 00 00 01 0b 00 00 69 06 19 1e 9e 00 00

Audit Entry 6:

00 00 00 01 0b 00 00 c8 19 0c 1e 9e 00 00

Audit Entry 7:

00 00 00 01 0b 00 00 c9 0c 04 1e 9e 00 00

Audit Entry 8:

00 00 00 01 0b 00 00 54 04 05 26 65 00 00

X->H

Header:

ff b1 dd 15 10

AIB:

00 01 02 02 00 00

Info:

00 05 00 00 12 09

Audit Entry 0:

00 00 00 00 00 00 00 0b 00 00 56 00 16 1d cd 00 00 ff

Audit Entry 1:

00 00 00 00 00 00 00 0b 00 00 c9 16 06 1d cd 00 00 ff

Audit Entry 2:

00 00 00 00 00 00 01 0b 00 00 67 06 1a 1e 96 00 00 ff

Audit Entry 3:

00 00 00 00 00 00 01 0b 00 00 c8 1a 0b 1e 96 00 00 ff

Audit Entry 4:

00 00 00 00 00 00 01 0b 00 00 c8 0b 06 1e 96 00 00 ff

Audit Entry 5:

00 00 00 00 00 00 01 0b 00 00 69 06 19 1e 9e 00 00 ff

Audit Entry 6:

00 00 00 00 00 00 01 0b 00 00 c8 19 0c 1e 9e 00 00 ff

Audit Entry 7:

00 00 00 00 00 00 01 0b 00 00 c9 0c 04 1e 9e 00 00 ff

Audit Entry 8:

00 00 00 00 00 00 01 0b 00 00 54 04 05 26 65 00 00 ff

The following shows the response in the format of the API message table. The data for the first audit entry only is detailed.

RESPONSE		
Byte	Field Description	Value and Indication
0	Frame	0xFE
1, 2	Length (MSB, LSB)	0x0092
3, 4	Message Type (MSB, LSB)	0x00DD (<i>PPL Audit Query</i>)
5	Reserved	0x00
6	Sequence #	0x15
7	Logical Node ID	0xFF
8, 9	Status (MSB, LSB)	0x0010 (Positive ACK)
10	AIB	0x00 (Single Entity)
11	# of Address Elements	0x01
12	Address Type	0x0D (Channel)
13	Data Length	0x03
14, 15	Logical Span (MSB, LSB)	0x0000 (Span 0)
16	Channel	0x00 (Channel 0)
17, 18	PPL Component ID (MSB, LSB)	0x0005 (ISDN L3P Call Reference)
19	Audit Type	0x00 (PPL State Transition Audit)
20	Audit Block Number	0x00 (Block 0)
21	Audit Entry Length	0x0F (15)
22	Number Of Audit Entries	0x09 (9)
23	Audit Entry 0 Data[0] Reserved	0x00
24	Data[1] Reserved	0x00
25, 26	Data[2,3] Call Reference (MSB, LSB)	0x0000
27	Data[4] PPL Protocol ID	0x0B (Protocol ID 11)
28	Data[5] State Status	0x00 (Normal)
29, 30	Data[6,7] PPL Event (MSB, LSB)	0x0056
31	Data[8] Initial State	0x00
32	Data[9] Next State	0x16
33, 34	Data[10,11] Timestamp (MSB, LSB)	0x1DCD
35, 36	Data[12,13] Error Status (MSB, LSB)	0x0000
37-50	Audit Entry 1: Data[0-13]	see trace for data
51-64	Audit Entry 2: Data[0-13]	see trace for data
65-78	Audit Entry 3: Data[0-13]	see trace for data
79-92	Audit Entry 4: Data[0-13]	see trace for data
93-106	Audit Entry 5: Data[0-13]	see trace for data
107-120	Audit Entry 6: Data[0-13]	see trace for data
121-134	Audit Entry 7: Data[0-13]	see trace for data
135-148	Audit Entry 8: Data[0-13]	see trace for data
149	Checksum	0xCS (not shown in trace)

PPL Configure 0x00D7

SwitchKit Name	PPLConfig
Type	EXS API and SwitchKit API message
Description	<p>PPL Configure 0x00D7</p> <p>This message is used to assign configuration data to a PPL component. Most PPL components have 200 bytes per channel available for data.</p> <p>NOTE: ANSI and ITU L3P CIC components have 600 bytes per channel.</p>
Sent by	Host

SwitchKit Code Configuration

```

PPLConfig (
    Node = integer,
    StartSpan = integer,
    StartChannel = integer,
    EndSpan = integer,
    Range = StartSpan:StartChan - EndSpan:EndChan,
    EndChannel = integer,
    Span = integer,
    Channel = integer,
    Subrate = integer,
    StackID = integer,
    LinkID = integer,
    Destination = integer,
    Route = integer,
    SSLGroupID = integer,
    SSLProfileID = integer,
    SSLServiceNumber = integer,
    V5ID = integer,
    RouterHandle = integer,
    ComponentID = integer,
    Entity = integer,
    ConfigData = byte array);

```

C Structure

```

typedef struct {
    UBYTE AddrInfo[30];
    unsigned short ComponentID;
    UBYTE Entity;
    unsigned short CSASlot;
    unsigned short Protocol;
    UBYTE ConfigData[220];
} XL_PPLConfig;

```

C++ Class

```

class XLC_PPLConfig : public XLC_OutboundMessage {

```

```

public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getStartSpan() const;
    void setStartSpan(XBYTE x);
    UBYTE getStartChannel() const;}
    void setStartChannel(UBYTE x);
    XBYTE getEndSpan() const;
    void setEndSpan(XBYTE x);
    UBYTE getEndChannel() const;
    void setEndChannel(UBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    UBYTE getLinkID() const;
    void setLinkID(UBYTE x)

    XBYTE getV5ID() const;
    void setV5ID(XBYTE x);
    XBYTE getRouterHandle() const;
    void setRouterHandle(unsigned short x);
    unsigned short getComponentID() const;
    void setComponentID(unsigned short x);
    UBYTE getEntity() const;
    void setEntity(UBYTE x);
    unsigned short getCSASlot() const;
    void setCSASlot(unsigned short x);
    unsigned short getProtocol() const;
    void setProtocol(unsigned short x) const;
    UBYTE *getConfigData() const;
    UBYTE *getConfigData();
    void setConfigData(UBYTE *x);
};
    
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00D7)	3, 4	Message Type (0x00D7)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID

PPL Configure 0x00D7

:	<u>AIB</u>	8, 9	Status MSB, LSB
	Address Method	10	Checksum
	Depends on the component being addressed.		
	See PPL Component Addressing in the <i>API Reference</i> .		
	Number of AEs to follow		
	AEs		
	Depends on the component being addressed.		
:	PPL Component ID MSB, LSB See PPL Component IDs in the <i>API Reference</i> .		
:	PPL Entity 0x01 PPL Configuration Bytes Block 1: 1-200 0x02 PPL Configuration Bytes Block 2: 201-400 0x03 PPL Configuration Bytes Block 3: 401-600 0x04 PPL Configuration Bytes Block 4: 601-800 0x05 PPL Configuration Bytes Block 5: 801-1000		
:	Configuration Data Number of Data Locations (n) For each location to be assigned data, indicate the byte to be configured (1–600) and the data. NOTE: If you are going to change the configuration of ISDN B Channels, do not send this message until you have sent the <i>ISDN Interface Configure</i> message.		
:	Location 1: Byte Number (in groups of 200, according to the value of the <i>PPL Entity</i> field).		
:	Location 1: Data		
:	Location n: Byte Number		
:	Location n: Data		
:	Checksum		

PPL Create 0x00D4

SwitchKit Name	PPLCreate
Type	EXS API and SwitchKit API message
Description	<p>PPL Create 0x00D4</p> <p>This message is used to create a PPL protocol from previously downloaded state/event and primitive tables. The host assigns the protocol ID and uses it to identify the protocol in subsequent messages.</p>
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short ComponentID;
    UBYTE ProtocolID;
    char ProtocolName[20];
    UBYTE NumTables;
    UBYTE TableData[229];
} XL_PPLCreate;
```

C++ Class

```
class XLC_PPLCreate : public XLC_OutboundMessage {
public:
    unsigned short GetComponentID() const;
    void setComponentID(unsigned short x);
    UBYTE getProtocolID() const;
    void setProtocolID(UBYTE x);
    const char *getProtocolName() const;
    void setProtocolName(const char *x);
    UBYTE getNumTables() const;
    void setNumTables(UBYTE x);
    const UBYTE *getTableData() const;
    UBYTE *getTableData();
    void setTableData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00D4)	3, 4	Message Type (0x00D4)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8, 9	PPL Component ID MSB, LSB See PPL Component IDs in the <i>API Reference</i> .	8, 9	Status MSB, LSB
10	PPL Protocol ID Number assigned to the protocol by the host (0x01–0x0A).	10	Checksum
11–30	PPL Protocol Name A NULL-terminated ASCII string of up to 20 characters describing the protocol (“ITU-T Protocol”). You must pad any remaining byte offsets up to 20 with 0x00.		
31	Number of Tables (n)		
:	Table 1: Type 0x01 Primitive Table 0x02 State/Event Table 0x03 Route Table 0x04 Resource Group Table		
:	Table 1: ID Number assigned to the table by the host (0x01–0x0A).		
:	Table n: Type		
:	Table n: ID		
:	Checksum		

PPL Data Query 0x00DE

SwitchKit Name	PPLDataQuery
Type	EXS API and SwitchKit API message
Description	<p>PPL Data Query 0x00DE</p> <p>Use this message to query the following PPL information for a channel or slot:</p> <ul style="list-style-type: none"> • PPL Configuration Bytes 1–200 • PPL Generic Timers • Current PPL Component State • PPL Table • Current Protocol ID • PPL Configuration Bytes 1–200 • PPL Configuration Bytes 201–400 • PPL Configuration Bytes 401–600 • PPL Configuration Bytes 601–800 • PPL Configuration Bytes 801–1000 • DTMF Timers
Sent by	Host
SwitchKit Code	<p>C Structure</p> <pre>typedef struct { UBYTE AddrInfo[30]; unsigned short ComponentID; UBYTE Entity; } XL_PPLDataQuery;</pre> <p>C Structure Response</p> <pre>typedef struct { unsigned short Status; UBYTE AddrInfo[30]; unsigned short ComponentID; UBYTE Entity; UBYTE ConfigData[218]; } XL_PPLDataQueryAck;</pre> <p>C++ Class</p> <pre>class XLC_PPLDataQuery : public XLC_OutboundMessage { public: const UBYTE *getAddrInfo() const; UBYTE *getAddrInfo(); void setAddrInfo(UBYTE *x); XBYTE getSpan() const;</pre>

```
void setSpan(XBYTE x);
UBYTE getChannel() const;
void setChannel(UBYTE x);
UBYTE getSlot() const;
void setSlot(UBYTE x);
XBYTE getV5ID() const;
void setV5ID(XBYTE x);
unsigned short getComponentID() const;
void setComponentID(unsigned short x);
UBYTE getEntity() const;
void setEntity(UBYTE x);
};
```

C++ Class Response

```
class XLC_PPLDataQueryAck : public XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    XBYTE getV5ID() const;
    void setV5ID(XBYTE x);
    unsigned short getComponentID() const;
    void setComponentID(unsigned short x);
    UBYTE getEntity() const;
    void setEntity(UBYTE x);
    const UBYTE *getConfigData() const;
    UBYTE *getConfigData();
    void setConfigData(UBYTE *x);
};
```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00DE)	3, 4	Message Type (0x00DE)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB (starting with byte 0)	8, 9	Status MSB, LSB
:	PPL Component ID MSB, LSB	10	AIB
:	PPL Configuration Entity	:	PPL Component ID MSB, LSB
:	Data [0] (if applicable)	:	PPL Configuration Entity
:	:	:	Response Data
:	Checksum	:	:
		:	Checksum

EXS API Hex Format - Detailed

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00DE)	3, 4	Message Type (0x00DE)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Element Depends on the component being addressed. See PPL Component Addressing in the <i>API Reference</i> . Number of AEs to follow AE Depends on the component being addressed.	8, 9	Status MSB, LSB
:	PPL Component ID MSB, LSB See PPL Component IDs in the <i>API Reference</i> .	10	AIB Same as message

PPL Data Query 0x00DE

:	<p>PPL Configuration Entity</p> <ul style="list-style-type: none"> 0x01 PPL Configuration Bytes 1–200 0x02 PPL Generic Timers 0x03 Current PPL Component State 0x04 PPL Table 0x05 Current PPL Component Extended State 0x06 Current Protocol ID 0x07 DTMF Timers 0x08 GK Status 0x10 PPL Configuration Bytes 1–200 0x11 PPL Configuration Bytes 201–400 0x12 PPL Configuration Bytes 401–600 0x13 PPL Configuration Bytes 601–800 0x14 PPL Configuration Bytes 801–1000 <p>NOTE: Use values 0x10–0x14 for ANSI and ITU L3P CIC components. Use the value 0x01 for all other components.</p>	:	<p>PPL Component ID MSB, LSB</p>
:	<p>Data</p> <p>This field applies to the PPL Table entity only.</p> <ul style="list-style-type: none"> 0x04 PPL Table <ul style="list-style-type: none"> Data[0] Table Type <ul style="list-style-type: none"> 0x03 Route Table 0x04 Resource Group Table Data[1] Table ID 0x01–0x0A Data[2, 3] Start of Range (MSB, LSB) <ul style="list-style-type: none"> Route Table ID: 1–1024 Resource Group ID: 1–256 Complete table: 0xFFFF 	:	<p>PPL Configuration Entity</p> <ul style="list-style-type: none"> 0x01 PPL Configuration Bytes 1–200 0x02 PPL Generic Timers <ul style="list-style-type: none"> If the Entity is 0x02 the data following are in two byte pairs starting with timer 1-100. Refer to appropriate <i>Developer's Guide</i> for the timers. 0x03 Current PPL Component State 0x04 PPL Table 0x06 Current Protocol ID 0x07 DTMF Timers 0x08 GK Status: <ul style="list-style-type: none"> 0x00 - undiscovered 0x01 - discovered, unregistered - response includes IP address (4 bytes) and port (2 bytes) 0x02 - registered - response includes IP address (4 bytes) and port (2 bytes) 0xFF - invalid 0x10 PPL Configuration Bytes 1–200 0x11 PPL Configuration Bytes 201–400 0x12 PPL Configuration Bytes 401–600 0x13 PPL Configuration Bytes 601–800 0x14 PPL Configuration Bytes 801–1000 <p>NOTE: Values 0x10–0x14 are used for ANSI and ITU L3P CIC components. The value 0x01 is used for all other components.</p>
:	<p>Checksum</p>		

:	<p>Response continued below.</p> <p>0x03 Current PPL Component State</p> <p style="padding-left: 40px;">0x00 OOS & Span Dead 0x02 TEI Assign Wait (Waiting for Terminal Endpoint Identifier assigned event.) 0x03 D Channel Alive Wait (Try to bring the link up, but network is not responding.) 0x04 Alive 0x05 Wait for Layer 1 Alive 0x06 Wait for D Dead Event (Waiting for D channel OOS event.)</p> <p>0x04 PPL Table</p> <p style="padding-left: 40px;">Data[0] Table Type Data [1] Table ID 0x01-0x0A 0x03 Route Table 0x04 Resource Group Table* NOTE: If Data[1] is 0x04 Resource Group Table, see farther down this table for information If Data[1] is 0x03 Route Table, Data[2-7] are as immediately follows:</p> <p style="padding-left: 40px;">Data[2, 3] Start of Range (MSB, LSB) Route Table ID: 0x01–0x400 Resource Group ID: 0x01–0x100 Complete Table: 0xFFFF</p> <p style="padding-left: 40px;">Data[4, 5] Number of Entries Data[6] Entry 1 Length Data[7] Entry 1 Data</p> <p style="padding-left: 80px;">Data[00 01] Route Number Data[02 03] Route Property 0x0001 Mask 0x0002 Range</p> <p style="padding-left: 80px;">Data[04 05] Route Group ID Data[06 07] Criteria Type</p>
---	--

Criteria Type All but Universal

Data[00 08] Criteria Data Length

Data[variable] Criteria data (actual .rte table data, "0" is 'don't care')

Byte[:]

See NOTE 1 for Data & Mask calculations

IF Route Property = 0x01,

Byte[a] = number of bytes in Data

Byte[b] = number of bytes in Mask

IF Route Property = 0x02,

Byte[x] = number of bytes in Start Range

Byte[y] = number of bytes in End Range

Byte[z] = number of bytes in Mask

Data[:] TLV Count (Number of TLV entries)

Data[:] TLV Tag See TLV Formats

Data[:] TLV Length

Data[:] TLV Value

:

Data[:] Entry N Length

Data[:] Entry N Data

Criteria Type Universal

Data[00 08] Criteria Data Length

Data[variable] Criteria data (actual .rte table data, "0" is 'don't care')

Byte[1] Univ data format

0x01 Raw Bytes

0x02 ASCII Bytes

0x03 BCD Data

Byte[2] Univ data offset

Byte[3-6] Reserved, should be 0

Byte[:]

See NOTE 2 for Data & Mask calculations

IF Route Property = 0x01

Byte[a] = number of bytes in Data

Byte[b] = number of bytes in Mask

IF Route Property = 0x02,

Byte[x] = number of bytes in Start Range

Byte[y] = number of bytes in End Range

Byte[z] = number of bytes in Mask

Data[:] TLV Count

Data[:] TLV Tag

See TLV Formats in the TLV chapter

Data[:] TLV Length

Data[:] TLV Value

:

Data[:] Entry N Length

Data[:] Entry N Data

	<p>*0x04 Resource Group Table NOTE: If Data[1] is 0x04 Resource Group Table, Data[2-7] are as follows:</p> <p>Data[2, 3] Start of Range (MSB, LSB) Route Table ID: 0x01–0x400 Resource Group ID: 0x01–0x100 Complete Table: 0xFFFF</p> <p>Data[4, 5] Number of Entries Data[6] Entry 1 Length Data[7] Entry 1 Data Data[00 01] Resource Group ID Data[02 03] Special Instruction (see special instruc. TLV [0x001D]) Data[variable] AIB See AIB Types :</p> <p>Data[:] Entry N Length Data[:] Entry N Data</p> <p>Data 0x05 Data[0, 1] Current State (MSB, LSB)</p>
:	Checksum

NOTE 1 The following information shows calculations for the *number* of bytes in the Data, Start/End Range, and Mask fields. The actual data within these bytes varies.

1. Convert Criteria Data Length to decimal value.
2. If Route Property = 0x01
 - a = number of bytes in Data = Criteria Data Length / 2
 - b = number of bytes in Mask = Criteria Data Length / 2
3. If Route Property = 0x02,
 - x = number of bytes in Start Range = Criteria Data Length / 3
 - y = number of bytes in End Range = Criteria Data Length / 3
 - z = number of bytes in Mask = Criteria Data Length / 3

NOTE 2 The following information shows calculations for the *number* of bytes in the Data, Start/End Range, and Mask fields. The actual data within these bytes varies.

1. Convert Criteria Data Length to decimal value.
2. If Route Property = 0x01
 - a = number of bytes in Data = (Criteria Data Length – 6) / 2
 - b = number of bytes in Mask = (Criteria Data Length – 6) / 2
3. If Route Property = 0x02,
 - x = number of bytes in Start Range = (Criteria Data Length – 6) / 3
 - y = number of bytes in End Range = (Criteria Data Length – 6) / 3

$$z = \text{number of bytes in Mask} = (\text{Criteria Data Length} - 6) / 3$$

If the response data cannot be sent by the CSP in one message, you must resend the message, incrementing the range, until a response status of 0xD7 is returned.

Criteria Type entries

0x0001 Address Digits 1
0x0002 Address Digits 2
0x0003 Address Digits 3
0x0004 Address Digits 4
0x0005 User-defined
0x0006 Route Group ID
0x0007 Incoming Span/Channel
0x0010 Incoming Resource Group
0x001A Incoming Node ID

Universal Protocol Data

A Universal Protocol is one of the options that can be used when defining a particular Criteria Type. You must distinguish between UDP and other possible entries to accurately show the data contained in the Route Tables.

0x09CE Build Index Table
0x2718 Calling Party Number (Connected Number)
0x27B0 RTP Payload Type
0x2792 Source IP Address
0x2792 Source IP Address
0x2793 Source RTP Port
0x2794 Destination IP Address
0x2795 Destination RTP Port
0x27B0 RTP Payload Type
0x27B1 RTP Payload Size

PPL Delete 0x00DA

SwitchKit Name PPLDelete

Type EXS API and SwitchKit API message

Description **PPL Delete 0x00DA**

This message allows the host to delete a protocol, a primitive table, or a state/event table. A primitive or state/event table is not be deleted if it is associated with a protocol in use. All channels must be out-of-service.

If the host sends this message to a CSP that has multiple line cards of the same type, the system deletes the protocol only if the protocol is not in use by every card. If the PPL protocol is in use by at least one card, the system returns a *Channel In Service* (0xD5) response and does not delete the channel.

You can delete all downloaded PPL components by specifying the PPL component 0xFFFF, which is valid for this message only. Using this value has the following effects:

- For E1 and T1 cards and matrices (L4PPL), the system deletes all downloaded PPLs and purges all channels using non-default protocols.
- For ISDN and SS7 cards, the system deletes all downloaded PPLs. If any channel (D/B channel, SS7 link, and so on) is using a non-default protocol *and* is in service, the system resets the card; however, if no channels are in service, the system does not reset the card.

Sent by Host

SwitchKit Code **Configuration**

```
PPLDelete (
    Node = integer,
    ComponentID = integer,
    Entity = integer,
    EntityID = integer);
```

C Structure

```
typedef struct {
    unsigned short ComponentID;
    UBYTE Entity;
    UBYTE EntityID;
} XL_PPLDelete;
```

C++ Class

```
class XLC_PPLDelete : public XLC_OutboundMessage {
public:
    unsigned short GetComponentID() const;
    void setComponentID(unsigned short x);
    UBYTE getEntity() const;
    void setEntity(UBYTE x);
    UBYTE getEntityID() const;
    void setEntityID(UBYTE x);
```

};

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x0009)	1, 2	Length (0x0007)
3, 4	Message Type (0x00DA)	3, 4	Message Type (0x00DA)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8, 9	PPL Component ID MSB, LSB See PPL Component IDs in the <i>API Reference</i> .	8, 9	Status MSB, LSB
		10	Checksum
10	PPL Entity 0x01 Primitive Table 0x02 State/Event Table 0x03 Route Table 0x04 Resource Group Table 0xFF Protocol		
11	PPL Entity ID 0x01–0x0A Host-defined table/protocol		
12	Checksum		

PPL Event Indication 0x0043

SwitchKit Name PPLEventIndication

Type EXS API and SwitchKit API message

Description **PPL Event Indication 0x0043**

This message is sent by a PPL component to report an event to the host with optional ICB data.

NOTE: All PPL Event Indication messages that originate with the CIC PPL component will have the new Virtual CIC AIB.

Sent by CSP

Example Message (Socket Log Output for SwitchKit)

In this example, the CSP informs the host of an incoming SIP REGISTER message:

```
00 0B 00 43 00 00 FF 7F 00 A7 00 0A 00
```

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    unsigned short ComponentID;
    unsigned short PPLEvent;
    UBYTE ICBCount;
    UBYTE Data[218];
} XL_PPPEventIndication;
```

C++ Class

```
class XLC_PPPEventIndication : public XLC_InboundMessage
{
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    UBYTE getLinkID() const;
    void setLinkID(UBYTE x);
    XBYTE getV5ID() const;
    void setV5ID(XBYTE x);
    unsigned short getComponentID() const;
    void setComponentID(unsigned short x);
    unsigned short getPPLEvent() const;
    void setPPLEvent(unsigned short x);
    UBYTE getICBCount() const;
    void setICBCount(UBYTE x);
```

```
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0005)
3, 4	Message Type (0x0043)	3, 4	Message Type (0x0043)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u> Address Method Depends on component being addressed. Number of AEs to follow AE The AE used depends on the component being addressed. See PPL Component Addressing in the <i>API Reference</i> .	8	Checksum
:	PPL Component ID MSB, LSB See PPL Component IDs in the <i>API Reference</i> .		
:	PPL Event MSB, LSB		
:	Number of ICBs to follow (Ignore this field if no ICBs in message.)		

:	<p>ICB</p> <p>0x02 Data ICBs</p> <p>0x10 ISDN Formatted IEs</p> <p>0x11 ISDN Raw IEs</p> <p>0x12 SS7 Parameters</p> <p>0x14 PPL Argument 2 Data</p> <p>0x15 DASS2/DPNSS Raw Data</p> <p>0x16 SS7 Protocol Violation Data</p> <p>0x1C SS7 TUP Formatted Fields</p> <p>0x1E Generic PPL</p> <p>0x1F SS7 Unformatted Raw Parameters</p> <p>0x20 SS7 SCCP Parameters</p> <p>0x21 SS7 TCAP Parameters</p> <p>0x22 Raw SS7 Data Parameters</p> <p>0x25 ISDN Segmented Message</p> <p>0x51 Soft Register All Switch-Initiated Messages</p> <p>0x5B IP Signaling Series 3 Card ID</p> <p>0x5D H.323 DTMF Signal Input Received</p> <p>0x5E H.323 Signal Update Input Received</p> <p>0x5F H.323 Alphanumeric Input Received</p> <p>0x62 Remote Endpoints UII Capabilities</p> <p>0x65 TCAP Primitive</p> <p>0x03 Extended Data ICBs</p> <p>0x002D Q.752 Parameter</p> <p>0x0012 SS7 Formatted Parameters</p> <p>0x0026 Channel Pad Value</p> <p>0x0029 V5 Formatted IEs</p> <p>0x0030 V5 Statistics Data</p> <p>0x0031 V5 Status Indication</p> <p>0x0033 NPDI Universal ICB</p> <p>0x0034 LAPD Frame</p> <p>0x0035 LAPD Status</p> <p>0x0065 TCAP Primitive</p>
	Refer to <i>Interworking TLVs (4-4)</i> if you have enabled interworking from the <i>VoIP Protocol Configure 0x00EE</i> message.
:	Checksum

Data For SS7 MTP Message Tracing:

ICB Data Length n (variable)

Data [0]First byte of MSU Data

Data [n]End of MSU Data

NOTE: If ICB length is more than 200 bytes the following applies.
For example: ICB Data Length is 450 bytes:

- a. Then the first frame will be sent with the first 200 bytes of MSU Data with PPL Event Indication as 0x20: "SS7 MTP Message Tracing Not Last".

- b. The second frame will be sent with the next 200 bytes of MSU Data with PPL Event Indication as 0x20: "MTP Message Tracing Not Last".
- c. The third frame will be sent with the next 50 bytes of MSU Data with PPL Event Indication as 0x21: "MTP Message Tracing Last".

The length and content of the data varies according to the ICB subtype. A list of valid ICBs appears in the *ICB* chapter.

Response Status

If the MSB of the Status field is 0x50 (NACK), the LSB indicates the NACK reason.

0x01 Invalid module type for DPC/Stack/CICcombination.

Example: Host has sent a *Service State Configure* message with invalid DPC/Stack/CIC.

0x02 Message received for unassigned CIC.

Example: Host has sent a *PPL Event Request* for unassigned CIC.

0x03 Only Extended API is supported for VCIC.

PPL Event Request 0x0044

SwitchKit Name PPLEventRequest

Type EXS API and SwitchKit API message

Description **PPL Event Request 0x0044**

This message is sent to initiate a host event on a PPL component with optional ICB data.

NOTE: All PPL Event Indication messages that originate with the CIC PPL component will have the Virtual CIC AIB.

Sent by Host

Sample Message In this sample, the host accepts a SIP registration request.

00 0B 00 44 00 00 FF 7F 00 A7 00 0A 00

SwitchKit Code **Configuration**

```
PPLEventRequest (
  Node = integer,
  Span = integer,
  Channel = integer,
  Subrate = integer,
  StackID = integer,
  LinkID = integer,
  Destination = integer,
  Route = integer,
  NumChannels = integer,
  V5ID = integer,
  TCAPStackID = integer,
  SSN = integer,
  ComponentID = integer,
  PPLEvent = integer,
  ICBCount = integer,
  Data = byte array);
```

C Structure

```
typedef struct {
  UBYTE AddrInfo[30];
  unsigned short ComponentID;
  unsigned short PPLEvent;
  UBYTE ICBCount;
  UBYTE Data[218];
} XL_PPLEventRequest;
```

C Structure Response

```
typedef struct {
  unsigned short Status;
  UBYTE reserved6[13];
```

```

    UBYTE Data[251];
} XL_PPLEventRequestAck;

```

C++ Class

```

class XLC_PPLEventRequest : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getNumChannels() const;
    void setNumChannels(UBYTE x);
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    UBYTE getLinkID() const;
    void setLinkID(UBYTE x);
    XBYTE getV5ID() const;
    void setV5ID(XBYTE x);
    unsigned short getComponentID() const;
    void setComponentID(unsigned short x);
    unsigned short getPPLEvent() const;
    void setPPLEvent(unsigned short x);
    UBYTE getICBCount() const;
    void setICBCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};

```

C++ Class Response

```

class XLC_PPLEventRequestAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0044)	3, 4	Message Type (0x0044)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u> Address Method Depends on component being addressed.	8, 9	Status (MSB, LSB) If the value of the response's <i>Status</i> field is anything other than 0x0010 (Positive Acknowledgment), the channel remains in its current state and the host can send subsequent messages such as Connect or Release Channel.
		10	Checksum
	Number of AEs to follow		
:	AE The AE used depends on the component being addressed. See PPL Component Addressing in the <i>API Reference</i> . PPL Component ID (MSB, LSB). See PPL Component IDs in the <i>API Reference</i> .		
:	PPL Event (MSB, LSB) See PPL Information in the <i>Developer's Guides</i> to find the PPL Events supported for each component. <i>Developers Guide: Internet Protocol (IP)</i> <i>Developers Guide: Line Cards</i> <i>Developers Guide: Common Channel Signaling</i> <i>Developers Guide: Overview</i>		
:	Number of ICBs to follow (Ignore this field if no ICBs in message).		
:	ICB Type 0x02 - Data 0x03 - Extended		

:	<p>0x02 Data ICBs</p> <ul style="list-style-type: none"> 0x10 ISDN Formatted IEs 0x11 ISDN Raw IEs 0x12 SS7 Parameters 0x14 PPL Argument 2 Data 0x15 DASS2/DPNSS Raw Data 0x16 SS7 Protocol Violation Data 0x1C SS7 TUP Formatted Fields 0x1E Generic PPL <p>The following TLVs are used to support the JT standard of Japanese telecommunications networks.</p> <ul style="list-style-type: none"> 0x11 Signaling Route Test Control Status 0x12 Signaling Route Test Control DPC 0x1F SS7 Unformatted Raw Parameters 0x20 SS7 SCCP Parameters 0x21 SS7 TCAP Parameters 0x22 Raw SS7 Data Parameters 0x25 ISDN Segmented Message 0x51 Soft Register All Switch-Initiated Messages 0x5B IP Signaling Series 3 Card ID 0x5D H.323 DTMF Signal Input Received 0x5E H.323 Signal Update Input Received 0x5F H.323 Alphanumeric Input Received 0x62 Remote Endpoints UII Capabilities 0x65 TCAP Primitive <p>0x03 Extended Data ICBs</p> <ul style="list-style-type: none"> 0x002D Q.752 Parameter 0x0012 SS7 Formatted Parameters 0x0030 V5 Statistics Data 0x0031 V5 Status Indication 0x0033 NPDI Universal ICB 0x0034 LAPD Frame 0x0035 LAPD Status 0x0065 TCAP Primitive
	<p>Refer to <i>Interworking TLVs (4-4)</i> if you have enabled interworking from the <i>VoIP Protocol Configure 0x00EE</i> message.</p>
:	Checksum

PPL Table Download 0x00D6

SwitchKit Name	PPLTableDownload
Type	EXS API and SwitchKit API message
Description	<p>PPL Table Download 0x00D6</p> <p>Use this message to download PPL tables that are subsequently used to create a PPL protocol.</p> <p>Before using this message, the host must first establish the table to be downloaded using the <i>PPL Table Initiate Download</i> message. The <i>PPL Table Download</i> message is then used to pass the table data to the CSP.</p> <p>If the table data exceeds the maximum length allowed for one message, you will have to use more than one of these messages. You must send them sequentially because the CSP maintains the offset into the table on successive commands.</p>
Sent by	Host
SwitchKit Code	<p>C Structure</p> <pre>typedef struct { unsigned short ComponentID; UBYTE TableType; UBYTE TableID; UBYTE Data[249]; } XL_PPLTableDownload;</pre> <p>C Structure Response</p> <pre>typedef struct { unsigned short Status; UBYTE DataSize; UBYTE ErrorType; UBYTE Data[250]; } XL_PPLTableDownloadAck;</pre> <p>C++ Class</p> <pre>class XLC_PPLTableDownload : public XLC_OutboundMessage { public: unsigned short GetComponentID() const; void setComponentID(unsigned short x); UBYTE getTableType() const; void setTableType(UBYTE x); UBYTE getTableID() const; void setTableID(UBYTE x); const UBYTE *getData() const; UBYTE *getData(); void setData(UBYTE *x); };</pre>

C++ Class Response

```
class XLC_PPLTableDownloadAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getDataSize() const;
    void setDataSize(UBYTE x);
    UBYTE getErrorType() const ;
    void setErrorType(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00D6)	3, 4	Message Type (0x00D6)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8, 9	PPL Component ID MSB, LSB See PPL Component IDs in the <i>API Reference</i> .	8, 9	Status MSB, LSB
10	Table Type (see formats below) 0x01 Primitive Table 0x02 Basic State/Event Table 0x03 Route Table 0x04 Resource Group Table 0x07 Extended State/Event Table	:	The following bytes apply if the Status is a NACK: "Table Does Not Validate" (0xA9). Otherwise, this field is 0x00, followed by the checksum. Error Data Length: The number of bytes of data to follow, including the <i>Error Type</i> field.

<p>11</p>	<p>Table ID Number assigned to table by host 0x01-0x0A</p>	<p>:</p>	<p>Error Type:</p> <p><u>General Errors</u> 0x01 Invalid Table Type 0x02 Invalid Checksum</p> <p><u>Primitive Table Errors</u> 0x03 Invalid Number of Atomic Functions In Primitive 0x04 Invalid Atomic Function ID 0x05 Invalid Primitive Table Size</p> <p><u>State/Event Table Errors</u> 0x06 Invalid Number of Events per State 0x07 Invalid Event ID (Exceeds Allowable Range) 0x08 Invalid Next State Type 0x09 Invalid State/Event Table Size</p> <p>Router Errors 0x0A Invalid route number 0x0B Invalid criteria type 0x0C Null TLV</p> <p>Resource Group Errors 0x0D Invalid number of AIBs in Resource Group 0x0E Invalid AIB in Resource Group 0x0F Incorrect number of address elements in Resource</p> <p>More Router Errors 0x10 Router Table Full 0x11 Router Table validation error</p> <p>0x0A Invalid route number 0x0B Invalid criteria type 0x0C Null TLV 0x0D Invalid number of AIBs in Resource Group 0x0D Invalid AIB in Resource Group 0x0F Incorrect number of address elements in Resource Group</p> <p>0x10 Router Table Full 0x11 Router Table validation error 0x12 Invalid Route Group Id for Criteria</p>
-----------	--	----------	---

12	Table Data - See tables below this one.	:	<p>Error Data[0] The data is dependent on the value on the <i>Error Type</i> field.</p> <p><u>(0x01) Invalid Table Type</u> Reported via a <i>Status</i> code.</p> <p><u>(0x02) Invalid Checksum</u> Bytes 0–3 Checksum</p> <p><u>(0x03) Invalid Number Of Atomic Functions</u> Bytes 0, 1 Primitive ID Byte 2 Number of Atomic Functions</p> <p><u>(0x04) Invalid Atomic Function ID</u> Bytes 0, 1 Primitive ID Bytes 2, 3 Atomic Function ID</p> <p><u>(0x05) Invalid Primitive Table Size</u> Bytes 0, 1 Primitive ID Bytes 2 Number of Atomic Functions</p> <p><u>(0x06) Invalid Number of Events Per State</u> Bytes 0, 1 State ID Bytes 2, 3 Number of events</p> <p><u>(0x07) Invalid Event ID</u> Bytes 0, 1 State ID Bytes 2, 3 Event ID</p> <p><u>(0x08) Invalid Next State Type</u> Bytes 0, 1 State ID Byte 2 Next State Type</p> <p><u>(0x09) Invalid State/Event Table Size</u> Bytes 0, 1 State ID Bytes 2, 3 Number of Events</p>
----	---	---	--

			(0x0A) Invalid route number Bytes 0, 1 Route Number
			(0x0B) Invalid criteria type Bytes 0, 1 Entry Data TLV Count
			(0x0C) Null TLV Bytes 0, 1 Entry Data TLV Count
			(0x0D) Invalid number of AIBs in Resource Group Bytes 0, 1 Resource Group ID
			(0x0E) Invalid AIB in Resource Group Bytes 0, 1 Resource Group ID
			(0x0F) Incorrect number of address elements in Resource Bytes 0, 1 Resource Group ID
			(0x10) Router Table Full Byte 0 Reserved Byte 1 Table ID
			(0x11) Router Table validation error Byte 0 Reserved Byte 1 Table ID
			(0x12) Invalid Route Group Id for Criteria Byte 0 Reserved
N + 3	Checksum	N+3	Checksum

Primitive Table Format (0x01)

A Primitive Table is a set of primitives used in conjunction with a State/Event Table to create a PPL protocol. Each Primitive Table can contain up to 400 primitives. The Primitive Table is assigned to the protocol when the protocol is created with the *PPL Create* message.

The Primitive Table format is shown in the table below.

Byte	Field Description	Definition
Message Header		
8, 9	PPL Component ID (MSB, LSB)	See list of Component IDs in this document.
10	Table Type	0x01 Primitive Table
11	Table ID	Assigned by host.
Table Data		
12, 13	Data[0,1] Primitive ID (MSB, LSB)	Identifies each primitive in the primitive table. Each primitive in the table must be unique, and in the range from 0 to 199. The primitives in the table must be ordered lowest number to highest (numbers may be skipped).
14	Data[2] Number of AFs	The number of AFs used to create the primitive (1 to 10).

15, 16	Data[3,4] AF: ID (MSB, LSB)	The AF ID specifies the predefined function to be invoked (in the range supplied by Dialogic).
17, 18	Data[5,6] Arg 1 Value (MSB, LSB)	The first argument value to pass to the AF.
19, 20	Data[7,8] Arg 2 Value (MSB, LSB)	The second argument value to pass to the AF.
Repeat Data[3-8] for each AF in the primitive.		
:	Checksum	

Basic State/Event Table Format (0x02)

A Basic State/Event table is a group of valid events that pertain to a given state, the primitives that get invoked upon the occurrence the event, and the next state to transition into in response to the event(s). State/event tables allow for up to 100 states and up to 20 events per state. The Primitive Table is assigned to the protocol when the protocol is created with the *PPL Create* message.

The Basic State/Event Table format is shown in the table below.

Byte	Field Description	Definition
Message Header		
8, 9	PPL Component ID (MSB, LSB)	See list of Component IDs in this document.
10	Table Type	0x02 Basic State/Event Table
11	Table ID	Assigned by host
Table Data		
12	Data[0] Table Data Initial State Number	Identifies the current state that the protocol is in. States must be specified in order from 1 to 199. State numbers can be skipped. The initial State Type is assumed to be Normal.
13	Data[1] Number of Events	The number of valid events accepted by the state. Each state can accept up to 120 events. For best performance, events most likely to occur should be placed at the beginning of the list.
14, 15	Data[2, 3] Event 1 # (MSB, LSB)	See the for event data.
16, 17	Data[4, 5] Event Data 1 Value (MSB, LSB)	
18, 19	Data[6, 7] Event Data 2 Value (MSB, LSB)	
20, 21	Data[8, 9] Primitive ID (MSB, LSB)	The primitive invoked when the specified event occurs based on the Primitive Table associated with the event.
22	Data[10] Next State Number	The state that the protocol should enter in response to the indicated event(s).
23	Data[11] Next State Type	0x01 Normal/Wait 0x02 Internal
Repeat Data[2-11] for each event associated with the state.		
:	Checksum	

Route Table (0x03) The Route Table format is shown in the table below.

Byte	Field Description	Definition
Message Header		
8, 9	PPL Component ID (MSB, LSB)	See PPL Information chapter in <i>API Reference</i> .
10	Table Type	0x03 Route Table
11	Table ID	Assigned by host
Table Data		
12, 13	Data[0, 1] Route Number	Index into the Route Table
14	Data[2] Route Property	This field is a bit mask indicating the details about the format of the remaining message bytes. <u>Bit</u> 0-4 Not used 5 Force the storage of entry in Sequential Linked List. If not set, use the default method of storage. 6 If set, use exact matching. If cleared use default method of matching 7 Version of Route Table. Should always be set.
15	Data[3] Route Property - Addressing method	This field is a bit mask indicating the addressing method, either single or range. <u>Bit</u> 0 Single 1 Range
16, 17	Data[2, 3] Version	Currently version 0x0001.
18, 19	Data[4, 5] Route Group ID	A grouping of routes based on a defined criteria or other information.
20, 21	Data[6, 7] Route Criteria Type. See Route Criteria Type entries below.	Identifies the type of information defining the route.
22	Data[8] Criteria Data: Format	0x01 Data in raw hex bytes 0x03 BCD (All digit based routes are BCD.)
23, 24	Data[9, 10] Criteria Data: Length of Criteria Data	
25	Data[11..N] Criteria Data	See Route Criteria Type entries below for the definition of remaining data.
:	Data[N+0, N+1] Entry Data TLV Count	2 Bytes
:	Data[N+2, N+3] Entry Data TLV Length	2 Bytes
:	Data[N+4] Entry Data: TLVs	See Route Entry Data TLV entries below.
:	Checksum	

Route Criteria Type entries

- 0x0007 Incoming Span/Channel, Single/Range
- 0x0009 Terminating Span/Channel, Single/Range
- 0x0010 Incoming Resource Group, Single/Range
- 0x0015 Terminating Resource Group, Single/Range
- 0x0065 Physical VoIP Channel
- 0x0071 Virtual VoIP Channel
- 0x03E9 User-defined

NOTE: See Notes 1, 2 and 3 below to configure the following Route Criteria Type entries.

- 0x2710 Universal Protocol Data, Single/Range
- 0x2717 Called Party Number, Single/Range
- 0x2718 Calling Party Number, Single/Range
- 0x2792 Source IP Address, Single/Range
- 0x2794 Destination IP Address, Single/Range
- 0x27B0 RTP Payload Type, Single/Range
- 0x27B4 IP Address Criteria, Single/Range
- 0x2919 NPDI SIP to Username, Single/Range
- 0x291A NPDI SIP to Hostname, Single/Range
- 0x2919 NPDI SIP from Username, Single/Range
- 0x291A NPDI SIP from Hostname, Single/Range
- 0x2A0E Media Connection Address, Single/Range

Notes

1. The second data format field that occurs after the length can have the following values:
 - 0x01 - Data in Raw Hex Bytes
 - 0x02 - ASCII Characters
 - 0x03 - BCD (all digit-based routes are BCD)
 - 0x04 - Decimal

2. The length of this field is variable. It is equal to the Criteria Length minus the length of the Criteria Header information (for example, all fields after the length and not including the digits and mask fields) divided by the number of digits and mask fields included.

Below is an example of how the Single bit in the Route Property Mask is set (see field 15 in Route Table (0x03) above).

To compute the Length of Field which is equal to 7, do the following:

Criteria Length (20) subtract (-) Length of Criteria Header (6) and divide (/) by number of Digit and Mask fields (2)

```

2a 0e    \ ' Media Connection Criteria Type
03       \ ' Criteria Data Format: BCD
00 14    \ ' Length
03       \ ' Criteria Data Format: BCD
63 \ ' Offset
00 00 00 00 \ ' Reserved - set to 0x00 0x00 0x00 0x00
04 05 05 06 06 07 08 \ ' Digits string
0f 0f 0f 0f 0f 0f 0f \ ' Mask
    
```

Below is an example of how the Range bit in the Route Property Mask is set (see field 15 in Route Table (0x03) above).

To compute the Length of Field which is equal to 1, do the following:

Criteria Length (9) subtract (-) Length of Criteria Header (6) and divide (/) by number of Digit and Mask fields (3)

- 2a 0e \ ' Media Connection Criteria Type
- 01 \ ' Criteria Data Format
- 00 09 \ ' Length
- 04 \ ' Criteria Data Format: Decimal
- 2c \ ' Offset
- 00 00 00 00 \ ' Reserved
- 0a \ ' From digits
- cd \ ' To digits
- ff \ ' Mask

3. The Offset field represents the number of channels to offset outgoing channel from incoming channel.

- Route Entry TLV Type entries**
- 0x0009 Address Element Blocks
 - 0x0015 Resource Group ID
 - 0x0062 Resource Group Reorder Method
 - 0x0063 Route Number
 - 0x001B Resource Table ID
 - 0x001C Route Table ID
 - 0x0161 PAD Values
 - 0x03E9 User-defined

Resource Group Table (0x04) The format of the Resource Group Table is shown in the table below.

Byte	Field Description	Definition
Message Header		
8, 9	PPL Component ID (MSB, LSB)	See PPL Information chapter in <i>API Reference</i> .
10	Table Type	0x04 Resource Group Table
11	Table ID	Assigned by host
Table Data		
12, 13	Data[0, 1] Resource Group ID	
14, 15	Data[2, 3] Special Instructions	0x0001
16, 17	Data[4, 5]	Version Number should be 0x0001
18, 19	Data[6, 7] Number of AIBs	
20, 21	Data[8, 9] Length of AIBs	
22+	Data[:] AIB	See AIB Types
:	Data[:] Number of TLVs	The number of Tag/Length/Value entries to follow.

:	Data[:] Length of TLVs	
:	Data[:] TLVs	See TLV Types
:	Checksum	

Extended State/Event Table Format (0x07)

An Extended State/Event table is a group of valid events that pertain to a given state, the primitives that get invoked upon the occurrence the event, and the next state to transition into in response to the event(s). State/event tables allow for up to 100 states and up to 20 events per state. The Primitive Table is assigned to the protocol when the protocol is created with the *PPL Create* message.

The Extended State/Event Table format is shown in the table below.

Byte	Field Description	Definition
Message Header		
8, 9	PPL Component ID (MSB, LSB)	See list of Component IDs in this document.
10	Table Type	0x07 Extended State/Event Table
11	Table ID	Assigned by host
Table Data		
12	Data[0,1] Initial State Number	Identifies the current state that the protocol is in. States must be specified in order from 1 to 199. State numbers can be skipped. The initial State Type is assumed to be Normal.
13	Data[2] Number of Events	The number of valid events accepted by the state. Each state can accept up to 120 events. For best performance, events most likely to occur should be placed at the beginning of the list.
14, 15	Data[3,4] Event 1 # (MSB, LSB)	Bytes 14 and 15 are external processing events, such as, Alerting or Answer; or internal decision events. Bytes 16-19 are arguments that relate to events such as, filter timer IDs for line signaling events. See the <i>API Developer's Guide: PPL</i> for more details.
16, 17	Data[5, 6] Event Data 1 Value (MSB, LSB)	
18, 19	Data[7, 8] Event Data 2 Value (MSB, LSB)	
20, 21	Data[9, 10] Primitive ID (MSB, LSB)	The primitive invoked when the specified event occurs based on the Primitive Table associated with the event.
22	Data[11, 12] Next State Number	The state that the protocol should enter in response to the indicated event(s).
23	Data[13] Next State Type	0x01 Normal/Wait 0x02 Internal
Repeat Data[2-13] for each event associated with the state.		
:	Checksum	

PPL Protocol Query 0x00DF

SwitchKit Name PPLProtocolQuery

Type EXS API and SwitchKit API message

Description **PPL Protocol Query 0x00DF**

This message initiates a report from the CSP on the custom protocols (non-default) that have been downloaded to a PPL component. There are two methods.

The first indicates every table downloaded to a particular PPL component and its associated protocols, if any. This is Sent by setting the Entity field to 0x00 (All Tables).

The second indicates the name of a particular entity (table or protocol). This is initiated by setting the Entity field to either 0x01 (Primitive Table), 0x02 (State/Event Table), or 0x03 (Protocol), and specifying the ID in the Entity ID field.

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short ComponentID;
    UBYTE Entity;
    UBYTE EntityID;
} XL_PPLProtocolQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    unsigned short ComponentID;
    UBYTE Entity;
    UBYTE EntityID;
    UBYTE Data[247];
} XL_PPLProtocolQueryAck;
```

C++ Class

```
class XLC_PPLProtocolQuery : public XLC_OutboundMessage {
public:
    unsigned short GetComponentID() const;
    void setComponentID(unsigned short x);
    UBYTE getEntity() const;
    void setEntity(UBYTE x);
    UBYTE getEntityID() const;
    void setEntityID(UBYTE x);
};
```

C++ Class Response

```
class XLC_PPLProtocolQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    unsigned short getComponentID() const;
    void setComponentID(unsigned short x);
    UBYTE getEntity() const;
    void setEntity(UBYTE x);
    UBYTE getEntityID() const;
    void setEntityID(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00DF)	3, 4	Message Type (0x00DF)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8, 9	PPL Component ID MSB, LSB See PPL Component IDs in the <i>API Reference</i> .	8, 9	Status MSB, LSB
10	Entity 0x00 All Tables 0x01 Primitive Table 0x02 State/Event Table 0x03 Protocol 0x07 Extended State/Event Table	10, 11	PPL Component ID MSB, LSB See PPL Component IDs in the <i>API Reference</i> .
11	Entity ID The ID of the table or protocol to be queried. If the Entity is All Tables (0x00) set this field to 0x00.	12	Entity
12	Checksum	13	Entity ID

	Response continued below.
14	<p>Data[0] The data varies by the <i>Entity</i> being queried.</p> <p>0x00 All Tables Data[0] Number of Tables Data[1] Table Type 0x01 Primitive Table 0x02 State/Event Table 0x07 Extended State/Event Table Data[2]* Table ID (1-10) Data[3]* Associated Protocol (0xFF if none)</p> <p>Data[1-3] is repeated for each table</p> <p>All Others: Data[0] Entity Name A NULL-terminated ASCII string of up to 20 characters describing the protocol ("ITU-T Protocol"). Remaining byte offsets up to 20 are padded with 0x00.</p>
:	Checksum

Examples Assumptions

- Component: 0x12
- Downloaded Protocols
 02 cpc_CPG_T7
- Primitive Tables
 02 isup_cpc_itu_ppl_pr
- State Event Tables
 02 isup_cpc_itu_ppl_st

Query All Tables

Message

To query all tables, you would send the message as shown below, with the following values:

- Component ID: **00 12**
- Entity: All Tables (**00**)
- Entity ID: **00**

H->X

[00 09 00 df 00 00 ff 00 12 00 00]

Response

The CSP returns the following bytes indicating that component 12 has 2 tables:

- Primitive Table 02 (Protocol 2)
- State Event Table 02 (Protocol 2)

X->H

[00 12 00 df 00 00 01 00 10 00 12 00 00 **02 01 02 02 02 02**
02]

Query Primitive Table

Message

To retrieve the name of Primitive Table 2, you would send the following bytes:

- Entity **01** (Primitive Table)
- Entity ID **02**

H->X

[00 09 00 df 00 00 ff 00 12 **01 02**]

Response

The CSP responds with a string of ASCII characters indicating the protocol name: isup_cpc_itu_ppl_pr

X->H

[00 1f 00 df 00 00 01 00 10 00 12 01 02 **69 73 75 70 5f 63**
70 63 5f 69 74 75 5f 70 70 6c 5f 70 72 00]

Query Entities 0x02 (State/Event Table), 0x03 (Protocol), and 0x07 (Extended State/Event Table) would return similar results.

PPL Table Download Initiate 0x00D5

SwitchKit Name	PPLInitiateDownload
Type	EXS API and SwitchKit API message
Description	<p>PPL Table Download Initiate 0x00D5</p> <p>This message allows the host to initiate the download of a primitive table or a state/event table for use in a protocol.</p> <p>This message results in the CSP allocating space for the specified table, which is then downloaded using the <i>PPL Table Download</i> message. The host assigns a Table ID to identify the table in subsequent messages. The host must manage the use of all PPL Table IDs.</p>

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short ComponentID;
    UBYTE TableType;
    UBYTE TableID;
    int TableSize;
    char TableName[20];
} XL_PPLInitiateDownload;
```

C++ Class

```
class XLC_PPLInitiateDownload : public
    XLC_OutboundMessage {
public:
    unsigned short GetComponentID() const;
    void setComponentID(unsigned short x);
    UBYTE getTableType() const;
    void setTableType(UBYTE x);
    UBYTE getTableID() const;
    void setTableID(UBYTE x);
    int getTableSize() const;
    void setTableSize(int x);
    const char *getTableName() const;
    void setTableName(const char *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x0021)	1, 2	Length (0x0007)
3, 4	Message Type (0x00D5)	3, 4	Message Type (0x00D5)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8, 9	PPL Component ID MSB, LSB See PPL Component IDs in the <i>API Reference</i> .	8, 9	Status MSB, LSB
		10	Checksum
10	Table Type 0x01 Primitive Table 0x02 State/Event Table 0x03 Route Table 0x04 Resource Group Table 0x07 Extended State/Event Tables		
11	Table ID Number assigned to the table by the host (0x01–0x0A).		
12-15	Table Size (4 Bytes) (total bytes in table to download)		
16–35	Table Name This field is a NULL-terminated ASCII string of up to 20 characters describing the table. Remaining byte offsets up to 20 should be padded with 0x00.		
36	Checksum		

PPLTool

Type	SwitchKit API message
Purpose	Use the PPLTool message to send either a hex format configuration (*.cfg) or a *.rep file (generated from Dialogic's technical support) to a specified node.
Description	This message downloads <u>either</u> a *.rep file or a *.cfg file (but not both) to the switch.
Sent by	SwitchManager
Arguments	The following table shows the user modifiable arguments of this message:

Argument	Description
Node	The node ID the file is destined for.
cfgFile	The *.cfg filename of the file to parse and send. The maximum size of the cfgFile name is 160 bytes.
repFile	The *.rep filename of the file to parse and send.

Configuration To download a PPL rep file:
pplTool(
 Node = integer,
 repFile = quoted string);

To download a configuration file:
pplTool(
 Node = integer,
 cfgFile = quoted string);

C Structure typedef struct {
 char RepFile[100];
 char CfgFile[100];
 } *SK_PPLTool*;

C++ Class class *SKC_PPLTool* : public SKC_DummyMessage {
 public:
 const char *getRepFile() const;
 void setRepFile(const char *x);
 const char *getCfgFile() const;
 void setCfgFile(const char *x);
 };

PPL Timer Configure 0x00CF

SwitchKit Name PPLTimerConfig

Type EXS API and SwitchKit API message

Description **PPL Timer Configure 0x00CF**

This message allows the host to configure PPL timers.

The 100 generic timers are used as defined by the protocol tables. The protocol designer must specify which timers are used, and how. This message allows the timers to be adjusted without having to modify the PPL State/Event or Primitive Tables.

NOTE: This message does not support the older ST1LC line cards. The correct message to use for these older line cards is the *Filter/Timer Configure* message (0x12).

Sent by Host

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00CF)	3, 4	Message Type (0x00CF)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8, 9	Status, MSB, LSB
:			
:	PPL Component ID MSB, LSB	10	Checksum
:	PPL Timer Type		
:	PPL Timer ID		
:	PPL Timer Value MSB, LSB		
:	Checksum		

SwitchKit Code Configuration

```
PPLTimerConfig (
    Node = integer,
    StartSpan = integer,
    StartChannel = integer,
    EndSpan = integer,
    Range = StartSpan:StartChan - EndSpan:EndChan,
    EndChannel = integer,
```

```

Span = integer,
Channel = integer,
Subrate = integer,
StackID = integer,
LinkID = integer,
Destination = integer,
Route = integer,
SSLGroupID = integer,
SSLProfileID = integer,
SSLServiceNumber = integer,
V5ID = integer,
ComponentID = integer,
TimerType = integer,
TimerID = integer,
TimerValue = integer);

```

C Structure

```

typedef struct {
    UBYTE AddrInfo[30];
    unsigned short ComponentID;
    UBYTE TimerType;
    UBYTE TimerID;
    unsigned short TimerValue;
} XL_PPLTimerConfig;

```

C++ Class

```

class XLC_PPLTimerConfig : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getStartSpan() const;
    void setStartSpan(XBYTE x);
    UBYTE getStartChannel() const;
    void setStartChannel(UBYTE x);
    XBYTE getEndSpan() const;
    void setEndSpan(XBYTE x);
    UBYTE getEndChannel() const;
    void setEndChannel(UBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    XBYTE getV5ID() const;
    void setV5ID(XBYTE x);
    unsigned short getComponentID() const;
    void setComponentID(unsigned short x);
    UBYTE getTimerType() const;
    void setTimerType(UBYTE x);
    UBYTE getTimerID() const;
    void setTimerID(UBYTE x);}
    unsigned short getTimerValue() const;
    void setTimerValue(unsigned short x);
};

```

**EXS API Hex Format -
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00CF)	3, 4	Message Type (0x00CF)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB	8, 9	Status, MSB, LSB
	Address Method Depends on component being addressed.		
	Number of AEs to follow	10	Checksum
	AE The AE used depends on the component being addressed. See PPL Component Addressing in the <i>API Reference</i> .		
:	PPL Component ID (MSB, LSB) See PPL Component IDs in the <i>API Reference</i> .		
:	PPL Timer Type 0x01 Generic Protocol Timer 0x02 R2 Signaling Timer 0x03 DTMF Signaling Timer 0x04 MFR1 Signaling Timer 0x05 Dial Pulse Timer		

:	<p>PPL Timer ID The specific timer according to its type. The meaning of the time ID depends on the value of the <i>PPL Timer Type</i> field. Valid entries for this field are listed below by <i>PPL Timer Type</i> field value. The default durations for timers are shown in parenthesis in 10 ms units.</p> <p>0x01 Generic Protocol Timer IDs Refer to Transmit Signaling Timers in the Timers and Filters chapter for the range of values.</p> <p>0x02 R2 Signaling Timer</p> <ul style="list-style-type: none"> 0x01 Pulse Duration (15) The length of time to transmit a pulsed R2 signal. 0x02 Pulse Delay (15) The minimum delay between the end of transmission of the last R2 signal of the compelled cycle and the start of transmission of a pulsed signal. 0x03 Minimum Valid Duration (7) The minimum length of time a signal must be received before it is declared a valid signal. 0x04 Maximum Tone State Change (2000) ñ the length of time to scan a receive R2 signal state change. <p>0x03 DTMF Signaling Timer</p> <ul style="list-style-type: none"> 0x01 DTMF Transmit Duration (6) 0x02 DTMF Transmit Inter-digit Duration (6) 0x03 DTMF Maximum Receive 1st Digit Detect (2000) 0x04 DTMF Maximum Receive Inter-digit Timeout (2000) 0x05 DTMF Minimum Receive Digit Duration (4) 0x06 DTMF Minimum Receive Inter-digit Timeout (4) <p>0x04 MFR1 Signaling Timer</p> <ul style="list-style-type: none"> 0x01 MFR1 Transmit KP Duration (10) 0x02 MFR1 Transmit Digit Duration (6) 0x03 MFR1 Transmit Inter-digit Duration (6) 0x04 MFR1 Minimum Receive KP Duration (5) 0x05 MFR1 Minimum Receive Digit Duration (3) 0x06 MFR1 Maximum Receive KP Detect (1000) 0x07 MFR1 Minimum Receive Inter-digit Timeout (3) 0x08 MFR1 Maximum Receive Inter-digit Timeout (1000)
---	---

PPL Timer Configure 0x00CF

:	<p>0x05 Dial Pulse Signaling Timer</p> <p>0x01 Dial Pulse Transmit Make Duration (4)</p> <p>0x02 Dial Pulse Transmit Break Duration (6)</p> <p>0x03 Dial Pulse Transmit Inter-digit Timeout (60)</p> <p>0x04 Dial Pulse Receive Minimum Make Duration (3)</p> <p>0x05 Dial Pulse Receive Maximum Make Duration (10)</p> <p>0x06 Dial Pulse Receive Minimum Break Duration (4)</p> <p>0x07 Dial Pulse Receive Maximum Break Duration (10)</p> <p>0x08 Dial Pulse Receive Maximum First Digit Detect (2000)</p> <p>0x09 Dial Pulse Receive Maximum Inter-digit Timeout (2000)</p>
:	<p>PPL Timer Value MSB, LSB</p> <p>Timers are in 10 millisecond units. See the <i>Timer Conversion Table (7-2)</i>.</p> <p>Changing the Connection Type with the ISDN Interface Configure message sets all B channel configuration parameters back to their defaults. If you need to change the default ISDN PPL configuration bytes or PPL timers you should first change the Connection Type.</p> <p>Note: For Generic Protocol Timer ID 0x04, the PPL Timer Value the preferred value is 0x0078.</p>
:	Checksum

PPL Transmit Signal Configure 0x00D2

SwitchKit Name	PPLTransmitSignalConfig
Type	EXS API and SwitchKit API message
Description	<p>PPL Transmit Signal Configure 0x00D2</p> <p>This message allows the host to configure transmit line signaling on channels with a trunk type of PPL.</p> <p>The idle signal is used by the internal out-of-service protocol to transmit idle code when transitioning to the primary protocol. The out-of-service signal is used when transitioning to the out-of-service protocol.</p>
Sent by	Host

SwitchKit Code Configuration

```
PPLTransmitSignalConfig (
    Node = integer,
    StartSpan = integer,
    StartChannel = integer,
    EndSpan = integer,
    Range = StartSpan:StartChan - EndSpan:EndChan,
    EndChannel = integer,
    ComponentID = integer,
    ProtocolSignalingState = integer,
    SignalingBits = integer);
```

C Structure

```
typedef struct {
    unsigned short StartSpan;
    UBYTE StartChannel;
    unsigned short EndSpan;
    UBYTE EndChannel;
    unsigned short ComponentID;
    UBYTE ProtocolSignalingState;
    UBYTE SignalingBits;
} XL_PPLTransmitSignalConfig;
```

C++ Class

```
class XLC_PPLTransmitSignalConfig : public
    XLC_OutboundMessage {
public:
    unsigned short getStartSpan() const;
    void setStartSpan(unsigned short x);
    UBYTE getStartChannel() const;
    void setStartChannel(UBYTE x);
    unsigned short getEndSpan() const;
    void setEndSpan(unsigned short x);
    UBYTE getEndChannel() const;
    void setEndChannel(UBYTE x);
    unsigned short GetComponentID() const;
```

PPL Transmit Signal Configure 0x00D2

```

void setComponentID(unsigned short x);
UBYTE getProtocolSignalingState() const;
void setProtocolSignalingState(UBYTE x);
UBYTE getSignalingBits() const;
void setSignalingBits(UBYTE x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00D2)	3, 4	Message Type (0x00D2)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	<u>AIB</u> Address Method Depends on component being addressed.	8, 9	Status MSB, LSB
	Number of AEs to follow	10	Checksum
	AE The AE used depends on the component being addressed. See PPL Component Addressing in the <i>API Reference</i> .		
:	PPL Component ID (MSB, LSB) See PPL Component IDs in the <i>API Reference</i> .		

PPL Transmit Signal Configure 0x00D2

	<p>Protocol Signaling State</p> <p>0x01 Out of Service 0x02 Idle 0x03 Break 0x04 Make</p> <p>(Break and Make are for Rotary/Dial Pulse/Decadics systems only.)</p>										
:	<p>Signaling Bits</p> <p>This field is a bit mask. The signaling bit pattern varies by PPL component:</p> <p>0x0001 E1 PPL</p> <table border="0"> <thead> <tr> <th><u>State</u></th> <th><u>Bit Pattern (0000ABCD)</u></th> </tr> </thead> <tbody> <tr> <td>Out of Service</td> <td>0000 1101 (0x0D)</td> </tr> <tr> <td>Idle</td> <td>0000 1001 (0x09)</td> </tr> <tr> <td>Break</td> <td>0000 1001 (0x09)</td> </tr> <tr> <td>Make</td> <td>0000 0001 (0x01)</td> </tr> </tbody> </table> <p>(Break and Make are for Rotary/Dial Pulse/Decadics systems only.)</p> <p>0x0003 T1 PPL</p> <p>Refer to <i>Table Configuration Bytes for T1 Component (2-4)</i> and <i>Table Default T1 Configuration Bytes, by trunk type</i> in the <i>Developer's Guide: Line Cards</i>.</p>	<u>State</u>	<u>Bit Pattern (0000ABCD)</u>	Out of Service	0000 1101 (0x0D)	Idle	0000 1001 (0x09)	Break	0000 1001 (0x09)	Make	0000 0001 (0x01)
<u>State</u>	<u>Bit Pattern (0000ABCD)</u>										
Out of Service	0000 1101 (0x0D)										
Idle	0000 1001 (0x09)										
Break	0000 1001 (0x09)										
Make	0000 0001 (0x01)										
:	Checksum										

NOTE: There are no default parameters for the E-ONE, T-ONE, or J-ONE line cards.

Product License Download 0x0079

SwitchKit Name ProductLicenseDownload

Type EXS API and SwitchKit API message

Description **Product License Download 0x0079**

This message is used to download one software key. Regardless of how a software or hardware module has been locked, you unlock all functionality with this message.

This message provides the mechanism for a host application to send the software feature or hardware upgrade license codes to the CSP. In a multi-node system, any node will accept and recognize these software feature licenses.

NOTES: For *SwitchKit*, when resetting a card such as, a T_ONE, E-ONE or J-ONE, a *ProductLicenseDownload* message is sent by SwitchManager so that the licensed hardware is available after a card reset. If the licensed card is permanently removed and the corresponding *ProductLicenseDownload* is not removed from the configuration, then any subsequent resets of matching card types will cause the *ProductLicenseDownload* to be NACKed.

Sent by Host

Example Message (Socket Log Output for SwitchKit)

The following socket log output/example message enables SIP software.
00 19 00 79 00 00 ff 01 02 24 10 30 38 42 50 43 4b 4b 48 49 42 32 34 54 50 4f 4b

SwitchKit Code **Configuration**

```
ProductLicenseDownload (
    Node = integer,
    ICBCount = integer,
    ICBData = byte array);
```

C Structure

```
typedef struct {
    UBYTE ICBCount;
    UBYTE ICBData[252];
} XL_ProductLicenseDownload;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE Slot;
} XL_ProductLicenseDownloadAck;
```

C++ Class

```
class XLC_ProductLicenseDownload : public
    XLC_OutboundMessage {
public:
    UBYTE getICBCount() const;
    void setICBCount(UBYTE x);
    const UBYTE *getICBData() const;
    UBYTE *getICBData();
    void setICBData(UBYTE *x);
};
```

C++ Class Response

```
class XLC_ProductLicenseDownloadAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x0079)	3, 4	Message Type (0x0079)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Number of ICBs to follow	8, 9	<u>Status MSB:</u> 0x4D If the MSB of the Status field is 0x4D (NACK), the LSB indicates the NACK reason. Response continued below.
9	ICB (Data Type) 0x24 Product License		
:	Checksum		

8, 9 (cont)	<p><u>Status LSB:</u></p> <p>0x01 Software Key Format Not Valid The first two bytes of the Product License field are not defined key types, or the product license contains invalid data.</p> <p>0x02 Data Decrypted Not Valid A serial number decrypted from a Product License cannot be matched against an existing serial number on the CSP.</p> <p>0x03 Product License - Insufficient Licensed Resources</p> <p>0x3D Product License - Invalid ICB Data</p> <p>0x74 Invalid Card Type</p> <p>0xAA Insufficient Hardware</p>
10	<p>Data</p> <p>For hardware upgrades, this field provides the slot number of the card that has been updated.</p> <p>For software module upgrades, this field is always "0".</p>
11	Checksum

Product License Query 0x007A

SwitchKit Name	ProductLicenseQuery
Type	EXS API and SwitchKit API message
Description	Product License Query 0x007A This message is used to provide information about product licenses that are currently installed on the CSP.
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {  
    } XL_ProductLicenseQuery;
```

C Structure Response

```
typedef struct {  
    unsigned short Status;  
    UBYTE NumLicenses;  
    UBYTE Data[250];  
} XL_ProductLicenseQueryAck;
```

C++ Class

```
class XLC_ProductLicenseQuery : public  
    XLC_OutboundMessage {  
public:  
    };
```

C++ Class Response

```
class XLC_ProductLicenseQueryAck : public  
    XLC_AcknowledgeMessage {  
public:  
    unsigned short getStatus() const;  
    void setStatus(unsigned short x);  
    UBYTE getNumLicenses() const;  
    void setNumLicenses(UBYTE x);  
    const UBYTE *getData() const;  
    UBYTE *getData();  
    void setData(UBYTE *x);  
    };
```

EXS API Hex Format

MESSAGE		RESPONSE	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x0005)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x007A)	3, 4	Message Type (0x007A)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Checksum		
Response continued below.			
8, 9	<p><u>Status MSB:</u> 0x4D If the MSB of the Status field is 0x4D (NACK), the LSB indicates the NACK reason.</p> <p><u>Status LSB:</u></p> <p>0x01 Software Key Format Not Valid The first two bytes of the Product License field are not defined key types, or if the product license contains invalid data.</p> <p>0x02 Data Decrypted Not Valid A serial number decrypted from a Product License cannot be matched against an existing serial number on the CSP.</p> <p>0x03 Product License - Insufficient Licensed Resources</p> <p>0x3D Product License - Invalid ICB Data</p> <p>0x74 Invalid Card Type</p> <p>0xAA Insufficient Hardware</p>		

10	Number of Licenses to follow
11	<p>Data[0] The data varies by the <i>Entity</i> being queried. The entity can be determined by Key Type. The values below are repeated for each Product License:</p> <p><u>Hardware Upgrades</u></p> <p>Data[0-1] Card Serial Number The serial number of the card that has been updated.</p> <p>Data[2] Slot Number The number of the slot of the card that has been updated.</p> <p>Data[3] Number of Entities The number of licensed spans for T-ONE or E-ONE cards, or the number of licensed links for SS7 cards.</p> <p>Data[4-5] Key Type The specific type of Product License. The value is listed in the Product License ICB with the corresponding License Type.</p> <p>Data[6-19] Encrypted Key A 14-byte hexadecimal value representing the encrypted license data</p> <p><u>Software Upgrades</u></p> <p>Data[0-1] Chassis Serial Number The serial number of the CSP backplane.</p> <p>Data[2] Slot Number The value of this field is for the chassis, and is always 0x13.</p> <p>Data[3] Number of Entities The value of this field is for the chassis, and is always 0x01.</p> <p>Data[4-5] Key Type The specific type of Product License.</p> <p>Data[6-19] Encrypted Key A 14-byte hexadecimal value representing the encrypted license data</p>
:	Checksum

Recorded Announcement File System Report 0x0119

SwitchKit Name RecAnnFSReport

Type EXS API and SwitchKit API message

Description **Recorded Announcement File System Report 0x0119**

This message reports information about the file system used by the VRAS SIMM's Flash memory on the DSP-ONE card. It also reports used, available, and deleted memory. To receive this message, the host must send a *Recorded Announcement File System Query*.

Sent by CSP

SwitchKit Code **C Structure**

```
typedef struct {  
    UBYTE Slot;  
    UBYTE SIMM;  
    UBYTE reserved19[28];  
    unsigned short RANCount;  
    unsigned short FSRevision;  
    int UsedMemory;  
    int DeletedMemory;  
    int AvailableMemory;  
    UBYTE Reserved;  
    UBYTE DefragmentationLevel;  
    UBYTE reserved65[205];  
} XL_RecAnnFSReport;
```

C++ Class

```
class XLC_RecAnnFSReport : public XLC_InboundMessage {  
public:  
  
    UBYTE getSlot() const;  
    void setSlot(UBYTE x);  
    UBYTE getSIMM() const;  
    void setSIMM(UBYTE x);  
    unsigned short getRANCount() const;  
    void setRANCount(unsigned short x);  
    unsigned short getFSRevision() const;  
    void setFSRevision(unsigned short x);  
    int getUsedMemory() const };  
    void setUsedMemory(int x);  
    int getDeletedMemory() const;  
    void setDeletedMemory(int x);  
    int getAvailableMemory() const;  
    void setAvailableMemory(int x);  
    UBYTE getReserved() const;  
    void setReserved(UBYTE x);  
    UBYTE getDefragmentationLevel() const;  
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x001D)	1, 2	Length (0x0007)
3, 4	Message Type (0x0119)	3, 4	Message Type (0x0119)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB Address Method 0x00 - Individual AEs Number of AEs to follow AE 0x12 DSP SIMM	8, 9	Status MSB, LSB
:		10	Checksum
14, 15	Number of RANs (MSB, LSB) Number of RANs on the VRAS SIMM, including those marked for deletion.		
16, 17	File System Revision (MSB, LSB) Revision of VRAS SIMM Flash File System used for storing RANs. 0x00 Revision 0: older, does not allow Single Message Deletion 0x01 Revision 1: newer, allows Single Message Deletion		
18 - 21	Used Memory (MSB, LSB) Number of 32-bit words of memory used by all RANs on the VRAS SIMM, including deleted RANs.		
22 - 25	Deleted Memory (MSB, LSB) Number of 32-bit words of memory used by deleted RANs on the VRAS SIMM		
26 - 29	Available Memory (MSB, LSB) Number of 32-bit words of memory available for use on the VRAS SIMM		
30	Reserved (0x00)		
31	Defragmentation Level Value calculated by the CSP, indicating the need to defragment the Flash memory. 0x00 Level 0: Lowest: No deleted memory. (Deleted memory = 0) 0x01 Level 1: Low Medium: Some deleted memory (Deleted memory > 0) 0x02 Level 2: High Medium (Available memory < 1/4 Total memory) and (Deleted memory > 1/4 Total memory) 0x03 Level 3: Highest. (Available memory < 1/4 Total memory) and (Deleted memory > 1/2 Total memory)		
32	Checksum		

Memory Field Information

The memory is reported in 32-bit words because that is the smallest memory storage unit. This means that a RAN six bytes long uses 2 words of memory for storing the RAN data.

Both File System revisions 0 and 1 use 16 words of memory for header information for each RAN stored.

File System revision 0 removes 16 words for each of the 2048 RANs (32768 words total) that could be stored on the VRAS SIMM from the available memory. So the total Available Memory for File System revision 0 is 0x5F8000. The Used Memory field counts only the memory words used for the RAN data, not the 16 words used for the RAN header.

File System revision 1 does not know the maximum number of RANs that may be stored on the VRAS SIMM, so the 16 words of header information for each RAN stored is added to the Used Memory for each RAN. Thus the total Available Memory for File System revision 1 is 0x600000. The deleted memory in the Defragmentation Level field also includes the 16 words for each deleted RAN.

Recorded Announcement File System Convert 0x0118

SwitchKit Name RecAnnFSConvert

Type EXS API and SwitchKit API message

Description **Recorded Announcement File System Convert 0x0118**

This message converts the file system used by the VRAS SIMM's Flash memory on the DSP-ONE card to the file system specified in the File System field. The VRAS SIMM must first be taken out of service with the *Service State Configure* message.

The host must wait for an *Alarm* message of “*Recorded Announcement File System Convert Success*” before bringing the VRAS SIMM in service. It may take up to five minutes for the alarm to be sent. If the conversion fails, the host receives an *Alarm* message of “*Recorded Announcement File System Convert Failure*” and the host should take action appropriate to the failure reason. The failure reason is in the Failure Code field of the *Alarm* message.

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {  
    UBYTE Slot;  
    UBYTE SIMM;  
    UBYTE reserved19[28];  
    unsigned short FSType;  
    UBYTE Reserved1;  
    UBYTE Reserved2;  
    UBYTE reserved51[219];  
} XL_RecAnnFSConvert;
```

C Structure Response

```
typedef struct {  
    unsigned short Status;  
    UBYTE Data[251];  
} XL_RecAnnFSConvertAck;
```

C++ Class

```
class XLC_RecAnnFSConvert : public  
XLC_OutboundMessage {  
public:  
  
    UBYTE getSlot() const;  
    void setSlot(UBYTE x);  
    UBYTE getSIMM() const;  
    void setSIMM(UBYTE x);  
    unsigned short getFSType() const;  
    void setFSType(unsigned short x);  
    UBYTE getReserved1() const;
```

```
void setReserved1(UBYTE x);
UBYTE getReserved2() const;
void setReserved2(UBYTE x);
};
```

C++ Class Response

```
class XLC_RecAnnFSConvertAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x000F)	1, 2	Length (0x0007)
3, 4	Message Type (0x0118)	3, 4	Message Type (0x0118)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs	8, 9	Status (MSB, LSB) 0x0B File System Conversion in Progress Trying to perform a VRAS function while the Flash memory on the VRAS SIMM is in the process of file system conversion 0x0C Defrag in Progress Trying to perform a VRAS function while the flash memory on the VRAS SIMM is defragmenting.
	Number of AEs to follow	10	Checksum
	AEs 0x12 DSP SIMM		
:	File System Type (MSB, LSB) 0x0000 Older Flash memory File System (does not permit single RAN deletion) 0x0001 Newer Flash memory File System (permits single RAN deletion)		
:	Reserved (0x0000)		
:	Checksum		

Recorded Announcement Single Delete 0x0117

SwitchKit Name RecAnnSingleDelete

Type EXS API and SwitchKit API message

Description **Recorded Announcement Single Delete 0x0117**

This message deletes a single Recorded Announcement (RAN), provided that the VRAS SIMM's Flash File System has been upgraded to revision 1 (0x0001). For example, deleting RAN ID 0x0001 on VRAS SIMM 0x02, Slot 0x00 deletes any instance of RAN ID 0x0001 stored on the DSP-ONE card in Slot 0x00, SIMM 0x02. But all other instances of RAN ID 0x0001 stored on a different DSP-ONE card, or on a different VRAS SIMM on the same card, remain unchanged.

This message can also be used just as the older message *Recorded Announcement Delete* (0x54) was used--to delete the VRA SIMM's entire Flash memory.

The host should wait for an *Alarm* message of "Single RAN Deleted on SIMM" or "VRAS Erase Complete" before performing another deletion or download of a RAN. This message may be sent to an In Service or Out of Service VRAS SIMM.

This message provides the following wildcards:

- Set the RAN ID field to 0xFFFF to delete all RANs on a Slot and VRAS SIMM (erase the VRAS SIMM's entire Flash memory). You **must** set the RAN ID to 0xFFFF if you are using the MFDSP card or you are using the DSP-ONE card with the older Flash File System, revision 0 (0x0000). When the RAN ID is 0xFFFF, a forced deletion is always performed (even currently-playing RANs are deleted).
- Set the VRAS SIMM field of the AIB to 0xFF to delete all instances of a RAN on all VRAS SIMMs.
- Set the Slot field of the AIB to 0xFF to delete all instances of a RAN on all slots.

You can combine the wildcards. For example, combining RAN ID 0xFFFF, VRAS SIMM 0xFF, and Slot 0xFF deletes **all** RANs stored on **all** VRAS SIMMs on **all** of the DSP cards (slots) in the chassis.

The Forced Flag field indicates whether the host deletes the RAN forcefully or gracefully. A forced RAN deletion cancels even currently-playing instances of that RAN. The forced deletion also cancels the playing of a RAN chain that needs the RAN ID on the DSP-ONE card. When the RAN ID is the wildcard (0xFFFF) a forced deletion is always performed.

A graceful RAN deletion is NACKed (and the RAN is not deleted) if the RAN is currently playing in Barge In mode, or if the RAN is needed for a current RAN chain or for a queued play request. If the RAN is currently playing, it is deleted only after it has finished playing. The graceful deletion is for single RAN IDs only.

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE Slot;
    UBYTE SIMM;
    UBYTE reserved19[28];
    unsigned short ID;
    UBYTE Reserved1;
    UBYTE Reserved2;
    UBYTE ForcedFlag;
    UBYTE reserved52[218];
} XL_RecAnnSingleDelete;
```

C++ Class

```
class XLC_RecAnnSingleDelete : public XLC_OutboundMessage
{
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x); }
    UBYTE getSIMM() const;
    void setSIMM(UBYTE x);
    unsigned short getID() const;
    void setID(unsigned short x);
    UBYTE getReserved1() const;
    void setReserved1(UBYTE x);
    UBYTE getReserved2() const;
    void setReserved2(UBYTE x);
    UBYTE getForcedFlag() const;
    void setForcedFlag(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x000F)	1, 2	Length (0x0007)
3, 4	Message Type (0x0117)	3, 4	Message Type (0x0117)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8, 9	Status (MSB, LSB)
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow		
	AE 0x12 DSP SIMM		
(14, 15)	RAN ID (MSB, LSB)		
	ID of the RAN to be deleted. To delete all RANs on the flash memory of this VRAS SIMM, use 0xFFFF.	(0x000B)	<u>File System Conversion in Progress</u> Trying to perform a VRAS function while the Flash memory on the VRAS SIMM is in the process of file system conversion.
16, 17	Reserved (0x0000)	(0x000C)	<u>Defrag in Progress</u> Trying to perform a VRAS function while the flash memory on the VRAS SIMM is in the process of defragmentation.
18	Forced Flag Valid only if deleting a Single RAN. 0x00 Graceful 0x01 Forceful	(0x000D)	<u>Deletion Already in Progress</u> Deletion of a single RAN ID has been requested for a RAN that is currently being deleted.
	Note: The forced request stops any playing of the RAN from the VRAS SIMM and then deletes the RAN. The graceful request allows any instance of the RAN playing from the VRAS SIMM to finish first, then deletes the RAN. The graceful delete request is NACKed if the DSP-ONE is playing a RAN in Barge-In mode, playing a chain that needs the RAN and this is the only VRAS SIMM on the DSP-ONE card that contains the RAN, or if there are currently requests waiting for a resource to play the RAN on this VRAS SIMM.	(0x000E)	<u>Invalid RAN File System</u> The Flash Memory File System does not support Single Message Deletion. Convert to the new file system with the API message, <i>Recorded Announcement File System Convert</i> .
		(0x000F)	<u>Graceful RAN Delete Denied</u> The RAN ID cannot be deleted gracefully, because it is being played in barge-in mode or it is needed for a pending play on this DSP-ONE card.
19	Checksum	10	Checksum

Recorded Announcement File System Defragment 0x0103

SwitchKit Name RecAnnFSDefrag

Type EXS API and SwitchKit API message

Description **Recorded Announcement File System Defragment 0x0103**

This message recovers unused memory in the file system used by the VRAS SIMM's Flash memory on the DSP-ONE card. It is used after single Recorded Announcements (RANs) have been marked for deletion using the *Recorded Announcement Single Delete* (0x0117) message. The VRAS SIMM must first be taken out of service with the *Service State Configure* message. The host should wait for an *Alarm* message of "Recorded Announcement Defragment Complete" before bringing the VRAS SIMM in service. If defragmentation fails, the host receives an *Alarm* message of "Recorded Announcement File Defragmentation Failure" and the host should take action appropriate to the failure reason. The failure reason is in the Failure Code field of the *Alarm* message.

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {  
    UBYTE Slot;  
    UBYTE SIMM;  
    UBYTE reserved19[251];  
} XL_RecAnnFSDefrag;
```

C Structure Response

```
typedef struct {  
    unsigned short Status;  
    UBYTE Data[251];  
} XL_RecAnnFSDefragAck;
```

C++ Class

```
class XLC_RecAnnFSDefrag : public  
XLC_OutboundMessage {  
public:  
    UBYTE getSlot() const;  
    void setSlot(UBYTE x);  
    UBYTE getSIMM() const;  
    void setSIMM(UBYTE x);  
};
```

C++ Class Response

```
class XLC_RecAnnFSDefragAck : public  
XLC_AcknowledgeMessage {  
public:
```

```

unsigned short getStatus() const;
void setStatus(unsigned short x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x000B)	1, 2	Length (0x0007)
3, 4	Message Type (0x0103)	3, 4	Message Type (0x0103)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB (Address Method 0x00 - Individual AEs	8, 9	Status MSB, LSB 0x0B File System Conversion in Progress Trying to perform a VRAS function while the Flash memory on the VRAS SIMM is in the process of file system conversion. 0x0C Defragmentation in Progress Trying to perform a VRAS function while the Flash memory on the VRAS SIMM is defragmenting. 0x0E Invalid VRAS File System The Flash Memory File System does not support defragmentation.
	Number of AEs	10	Checksum
	AE 0x12 DSP SIMM		
:	Checksum		

Recorded Announcement File System Query 0x0102

SwitchKit Name	RecAnnFSQuery
Type	EXS API and SwitchKit API message
Description	Recorded Announcement File System Query 0x0102 This message queries the file system used by the VRAS SIMM's Flash memory on the DSP-ONE card. In response to this message, the CSP sends a <i>Recorded Announcement File System Report</i> (0x0119).
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {  
    BaseFields Base;  
    UBYTE Slot;  
    UBYTE SIMM;  
    UBYTE reserved19[251];  
} XL_RecAnnFSQuery;
```

C Structure Response

```
typedef struct {  
    unsigned short Status;  
    UBYTE Data[251];  
} XL_RecAnnFSQueryAck;
```

C++ Class

```
class XLC_RecAnnFSQuery : public  
XLC_OutboundMessage {  
public:  
    UBYTE getSlot() const;  
    void setSlot(UBYTE x);  
    UBYTE getSIMM() const;  
    void setSIMM(UBYTE x);  
};
```

C++ Class Response

```
class XLC_RecAnnFSQueryAck : public  
XLC_AcknowledgeMessage {  
public:  
    unsigned short getStatus() const;  
    void setStatus(unsigned short x);  
    const UBYTE *getData() const;  
    UBYTE *getData();  
    void setData(UBYTE *x);  
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x000B)	1, 2	Length (0x0007)
3, 4	Message Type (0x0102)	3, 4	Message Type (0x0102)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs	8, 9	Status (MSB, LSB)
	Number of AEs to follow	10	Checksum
	AE 0x12 DSP SIMM		
:	Checksum		

Recorded Announcement Report 0x0058

SwitchKit Name RecAnnReport

Type EXS API and SwitchKit API message

Description **Recorded Announcement Report 0x0058**

This message is generated in response to a *Recorded Announcement Query* message sent by the host. It can contain information on up to 10 announcements.

Sent by CSP

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE Slot;
    UBYTE SIMM;
    UBYTE AnnounceCount;
    UBYTE Data[222];
} XL_RecAnnReport;
```

C++ Class

```
class XLC_RecAnnReport : public XLC_InboundMessage {
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getSIMM() const;
    void setSIMM(UBYTE x);
    UBYTE getAnnounceCount() const;
    void setAnnounceCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0005)
3, 4	Message Type (0x0058)	3, 4	Message Type (0x0058)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8	Checksum
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow		
	AE 0x12 DSP SIMM		
:	Announcement Count (n) 0x01–0x0A The number of announcements being reported in this message		
:	Announcement ID (MSB, LSB) The number the host has assigned to the announcement.		
:	Announcement Size (4 bytes) A long word containing the number of bytes in the announcement.		
:	Announcement Checksum (4 bytes) A long word containing the simple sum of all data bytes of the announcement.		
:	Announcement Format 0x00 8-bit PCM		
:	Announcement Encoding Format 0x00 A-law 0x01 μ -law		
:	Checksum		

NOTE: The *Announcement* fields are repeated for each announcement. The number of announcements is defined in the *Announcement Count* field.

Recorded Announcement Query 0x0057

SwitchKit Name RecAnnQuery

Type EXS API and SwitchKit API message

Description **Recorded Announcement Query 0x0057**

This message allows the host to query information about the announcements stored on a VRAS SIMM. The information is reported to the host via the *Recorded Announcement Report* message:

- If announcements are stored on the SIMM, the response to this message indicates the number of announcements stored on the specified SIMM.
- If no announcements are stored on the VRAS SIMM, the CSP does *not* send a report.

In the *Recorded Announcement Report* message, the host will receive the ID, size, checksum, format, and encoding format for each announcement stored on the specified SIMM.

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE Slot;
    UBYTE SIMM;
} XL_RecAnnQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE Slot;
    UBYTE SIMM;
    unsigned short AnnounceCount;
} XL_RecAnnQueryAck;
```

C++ Class

```
class XLC_RecAnnQuery : public XLC_OutboundMessage {
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getSIMM() const;
    void setSIMM(UBYTE x);
};
```

C++ Class Response

```
class XLC_RecAnnQueryAck : public XLC_AcknowledgeMessage
{
public:
    unsigned short getStatus() const;
```

```

void setStatus(unsigned short x);
UBYTE getSlot() const;
void setSlot(UBYTE x);
UBYTE getSIMM() const;
void setSIMM(UBYTE x);
unsigned short getAnnounceCount() const;
void setAnnounceCount(unsigned short x);
};
    
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x0057)	3, 4	Message Type (0x0057)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB)
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow	10	AIB (same as message)
	AE 0x12 DSP SIMM	:	Announcement Count (MSB, LSB) The total number of announcements on the specified VRAS SIMM.
:	Checksum	:	Checksum

Recorded Announcement Connect 0x0055

SwitchKit Name RecAnnConnect

Type EXS API and SwitchKit API message

Description **Recorded Announcement Connect 0x0055**

Use this message to connect a channel to one or a series of downloaded voice recorded announcements (RANs).

NOTE: You can use this message for the DSP Series 2 card, but only for RAN IDs below 4096. In CSPs using the DSP Series 2 card with the DSP-ONE card, the CSP Matrix Series 3 Card examines the announcement ID(s) and determines if there are any DSP-ONE cards that can play the entire list of announcements. If there is no DSP-ONE card that can do this, the request is passed to the DSP Series 2 card. Because of this, the DSP Series 2 card must either have all the same announcements loaded onto it as all the DSP-ONEs via TFTP, or all the announcements must be accessible via NFS.

Sent by Host

C Structure

```
typedef struct {
    unsigned short Span;
    UBYTE Channel;
    UBYTE Config;
    UBYTE Event;
    UBYTE Cnt;
    unsigned short ID1;
    unsigned short ID2;
    unsigned short ID3;
    unsigned short ID4;
    unsigned short ID5;
    unsigned short ID6;
    unsigned short ID7;
    unsigned short ID8;
    unsigned short ID9;
    unsigned short ID10;
    UBYTE OtherIDs[200];
} XL_RecAnnConnect;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    unsigned short Span;
    UBYTE Channel;
} XL_RecAnnConnectAck;
```

C++ Class

```
class XLC_RecAnnConnect : public XLC_OneChannelOutbound {
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
```

```
void setChannel(UBYTE x);
UBYTE getConfig() const;
void setConfig(UBYTE x);
UBYTE getEvent() const;
void setEvent(UBYTE x);
UBYTE getCnt() const;
void setCnt(UBYTE x);
unsigned short getID1() const;
void setID1(unsigned short x);
unsigned short getID2() const;
void setID2(unsigned short x);
unsigned short getID3() const;
void setID3(unsigned short x);
unsigned short getID4() const;
void setID4(unsigned short x);
unsigned short getID5() const;
void setID5(unsigned short x);
unsigned short getID6() const;
void setID6(unsigned short x);
unsigned short getID7() const;
void setID7(unsigned short x);
unsigned short getID8() const;
void setID8(unsigned short x);
unsigned short getID9() const;
void setID9(unsigned short x);
unsigned short getID10() const;
void setID10(unsigned short x);
const UBYTE *getOtherIDs() const;
UBYTE *getOtherIDs();
void setOtherIDs(UBYTE *x);
};
```

C++ Class Response

```
class XLC_RecAnnConnectAck : public
XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0055)	3, 4	Message Type (0x0055)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs		
	Number of AEs to follow		
	AE 0x0D Channel	8, 9	Status (MSB, LSB)
		10, 11	State (MSB, LSB) If the preceding Status field is a positive acknowledgement, this field does not apply.
		:	AIB (same as message)
		:	Checksum
:	Configuration Flag This field is a bit mask. <u>Bit</u> 0 If set, enables Barge-In feature. The channel connects to an announcement even if the announcement is in progress. This is valid only if the announcement count is 0x01. 1-7 Reserved (must be 0x00)		
:	Event Flag This field is a bit mask. <u>Bit</u> 0 - If set, a Call Processing Event of "Recorded Announcement Completed" (0x26) is generated when the announcement is complete. 1 - If set, a Call Processing Event of "Recorded Announcement Started" (0x25) is generated when the announcement starts. 2-7 - Reserved (Must be 0x00)		
:	Announcement Count (n) Number of announcements to follow. The host can chain together multiple announcements to be played out in sequence to a channel (0x01-0x40).		
:	Announcement ID [0, 1]		
:	Announcement ID [2, 3]		
:	:		
:	Announcement ID [x, y]		
:	Checksum		

Recorded Announcement Disconnect 0x0056

SwitchKit Name	RecAnnDisconnect
Type	EXS API and SwitchKit API message
Description	<p>Recorded Announcement Disconnect 0x0056</p> <p>This message allows the host to disconnect a channel from a voice recorded announcement.</p> <p>NOTE: Use the <i>RAN Disconnect</i> message to cancel only RAN Play requests made with the RAN Connect message, which can be used only for RAN IDs below 4096.</p>
Sent by	Host

SwitchKit Code C Structure

```
typedef struct {
    unsigned short Span;
    UBYTE Channel;
} XL_RecAnnDisconnect;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    unsigned short Span;
    UBYTE Channel;
} XL_RecAnnDisconnectAck;
```

C++ Class

```
class XLC_RecAnnDisconnect : public
    XLC_OneChannelOutbound {
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
};
```

C++ Class Response

```
class XLC_RecAnnDisconnectAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x0056)	3, 4	Message Type (0x0056)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8, 9	Status (MSB, LSB)
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow		
	AE 0x0D Channel		
:	Checksum	:	AIB (same as message)
		:	Checksum

This message always returns a positive acknowledgment (ACK).

Recorded Announcement Delete 0x0054

SwitchKit Name RecAnnDelete

Type EXS API and SwitchKit API message

Description **Recorded Announcement Delete 0x0054**

This message deletes *all* announcements stored on a VRAS SIMM. Deletion of a SIMM takes approximately 30 seconds. The host should wait for an *Alarm* message of “VRAS Erase Complete” before downloading a new announcement to a SIMM.

Sent by Host

SwitchKit Code **Configuration**

```
RecAnnDelete (
    Node = integer,
    Slot = integer,
    SIMM = integer,
    Reserved = integer,
    ForcedFlag = integer);
```

C Structure

```
typedef struct {
    UBYTE Slot;
    UBYTE SIMM;
    unsigned short Reserved;
    UBYTE ForcedFlag;
} XL_RecAnnDelete;
```

C++ Class

```
class XLC_RecAnnDelete : public XLC_OutboundMessage {
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getSIMM() const;
    void setSIMM(UBYTE x);
    unsigned short getReserved() const;
    void setReserved(unsigned short x);
    UBYTE getForcedFlag() const;
    void setForcedFlag(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0054)	3, 4	Message Type (0x0054)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs Number of AEs to follow AE 0x12 DSP SIMM	8, 9	Status (MSB, LSB) 0x0009 DSP Chip Dead When the DSP chip is taken out of service by the system, this response is returned in the NACK. 0x000B File System Conversion in Progress Trying to perform a VRAS function while the flash memory on the SIMM is in the process of file system conversion. 0x000C Defragmentation in Progress Trying to perform a VRAS function while the flash memory on the SIMM is in the process of defragmentation.
:	Reserved (3 bytes)	10	Checksum
:	Checksum		

Recorded Announcement Download 0x0053

SwitchKit Name RecAnnDownload

Type EXS API and SwitchKit API message

Description **Recorded Announcement Download 0x0053**

This message is used to download an announcement to VRAS SIMM and is no longer than 230 bytes. The RAN download is made up of one *RAN Download Initiate* message (0x0052) followed by many *RAN Download* messages (0x0053).

Sent by Host

Example Message (Socket Log Output for SwitchKit)

RAN Download Initiate (downloading RAN ID 0 to Slot 0 Simm 2):

```
00 17 00 52 00 00 ff 00 01 12 02 00 02 00 00 00 00 02 eb 00 01 1d d8 00 01
```

RAN Download (announcement data in bold):

```
00 f3 00 53 00 00 ff 00 01 12 02 00 02 00 00 ff 89 85
98 1b 0b 14 63 a2 ab e7 c5 9f 9b d3 12 09 17 9b 84 88
be 0b 05 17 a1 8e 97 ce 2d 43 c6 3b 1a 15 3c 90 87 92
1e 05 07 2f 8d 87 96 29 13 1b 4a bf 43 32 b9 95 8f af
10 05 0f a3 85 86 a8 10 09 17 b2 99 a1 cf e4 ab a7 3a
11 0d 27 91 84 8d 2b 06 06 22 94 8b 98 3f 1f 2c 6b 3e
1f 1e be 8f 8a 9f 13 03 0b b8 88 86 9e 19 0d 1a d2 ab
c3 4d b3 99 98 4e 0e 08 1a 95 83 89 52 0a 07 1d 9e 91
9d 7c 3c c5 b9 2e 13 13 5e 8d 86 96 18 04 09 4a 8c 89
9c 26 17 24 66 62 2c 2c ac 8f 8e bd 0d 04 11 9c 84 88
b7 0f 0b 1c ae 9e ad f3 ba 9f a3 2c 0d 0c 2f 8d 83 8f
20 06 08 2d 93 8d 9e 3a 28 44 da 2d 19 1c ae 8c 89 a6
0e 03 0d a9 88 89 a8 19 10 21 c9 ba 47 43 a8 93 97 35
0b 07 1e 90 83 8c 33 0a 0a 25 9e
```

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE Slot;
    UBYTE SIMM;
    unsigned short ID;
    UBYTE Data[221];
} XL_RecAnnDownload;
```

C++ Class

```
class XLC_RecAnnDownload : public XLC_OutboundMessage {
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getSIMM() const;
    void setSIMM(UBYTE x);
```

```

unsigned short getID() const;
void setID(unsigned short x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};
    
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0053)	3, 4	Message Type (0x0053)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Response Status (MSB, LSB) 0x0009 DSP Chip Dead When the DSP chip is taken out of service by the system, this response is returned in the NACK. 0x000B File System Conversion in Progress Trying to perform a VRAS function while the flash memory on the SIMM is in the process of file system conversion. 0x000C Defragmentation in Progress Trying to perform a VRAS function while the flash memory on the SIMM is in the process of defragmentation.
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow		
	AE 0x12 DSP SIMM	10	Checksum
:	Announcement ID (MSB, LSB) The number the host assigned to identify the announcement.		
:	Data		
:	Checksum		

Recorded Announcement Download Initiate 0x0052

SwitchKit Name RecAnnDownloadInit

Type EXS API and SwitchKit API message

Description **Recorded Announcement Download Initiate 0x0052**

This is the first message sent by the host to initiate the downloading of a recorded announcement to a VRAS SIMM. If multiple SIMMs are to have the same announcement, each SIMM must be downloaded separately. If you try to use this message for the DSP Series 2 card, the CSP Matrix Series 3 Card NACKs it.

For SwitchKit

This message downloads a voice file to a VRAS board on the CSP. The format and encoding correspond to the arguments of the RecAnnDownloadInitiate message. The slot and SIMM must refer to a valid VRAS SIMM, and the ID is used to reference this prompt from future messages. If the voice file is already present on the SIMM, the SwitchManager bypasses the downloading. If there is another voice file with that ID already on the SIMM, then the configuration fails. The VRAS SIMM must be cleared with a RecAnnDelete before the new voice file is downloaded.

Sent by Host (SwitchManager for SwitchKit)

Example Message (Socket Log Output for SwitchKit)

See *Recorded Announcement Download 0x0053* for the socket log output/example of the *Recorded Announcement Download Initiate* message and a *Recorded Announcement Download* message together.

SwitchKit Code **Configuration**

```
RecordedAnnouncement(  
    file = quoted string,  
    slot = integer,  
    SIMM = integer,  
    id = integer,  
    format = integer,  
    encoding = integer  
);
```

C Structure

```
typedef struct {  
    UBYTE Slot;  
    UBYTE SIMM;  
    unsigned short ID;  
    int Size;  
    int Checksum;  
    UBYTE Format;  
    UBYTE Encoding;  
} XL_RecAnnDownloadInit;
```

C++ Class

```
class XLC_RecAnnDownloadInit : public XLC_OutboundMessage
{
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getSIMM() const;
    void setSIMM(UBYTE x);
    unsigned short getID() const;
    void setID(unsigned short x);
    int getSize() const;
    void setSize(int x);
    int getChecksum() const;
    void setChecksum(int x);
    UBYTE getFormat() const;
    void setFormat(UBYTE x);
    UBYTE getEncoding() const;
    void setEncoding(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0052)	3, 4	Message Type (0x0052)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs	8, 9	Response Status (MSB, LSB) 0x000A Invalid RAN Size When the RAN_SIZE is larger than the permitted size of 16,777,215 bytes (0xFFFFFFFF), this response is returned in the negative acknowledgment to the <i>Recorded Announcement Download Initiate 0x0052</i> message.
	Number of AEs to follow		
	AE 0x12 DSP SIMM		
		10	Checksum
:	RAN ID (MSB, LSB)		
:	Announcement ID (MSB, LSB) A number the host assigns to identify the announcement (0x00-0xFFF).		
:	Announcement Size A long word (4 bytes) with the number of bytes in the announcement to be downloaded.		
:	Announcement Checksum A long word (4 bytes) containing the simple sum of all the data bytes of the announcement.		

Recorded Announcement Download Initiate
0x0052

:	Announcement Format 0x00 8-bit PCM
:	Encoding Format 0x00 A-law 0x01 μ -law
:	Checksum

Receive Signaling Configure 0x0015

SwitchKit Name ReceiveSignalingConfig

Type EXS API and SwitchKit API message

Description **Receive Signaling Configure 0x0015**

This message is used to define the signaling received by the CSP for non-standard trunk or line interfaces on the STILC card.

This message **cannot** be used with the T-ONE card. Line signaling for the T-ONE card is configured with T1 PPL Config Bytes 111–116 using the *PPL Configure* message.

NOTE: This message should not be used without first contacting Dialogic Technical Support.

Sent by Host

SwitchKit Code **Configuration**

```
ReceiveSignalingConfig (
    Node = integer,
    Range = StartSpan:StartChan - EndSpan:EndChan,
    SignallingType = integer,
    SignallingValue = integer,
    TransmissionMode = integer,
    BitMask = integer);
```

C Structure

```
typedef struct {
    unsigned short StartSpan;
    UBYTE StartChannel;
    unsigned short EndSpan;
    UBYTE EndChannel;
    UBYTE SignallingType;
    UBYTE SignallingValue;
    UBYTE TransmissionMode;
    UBYTE BitMask;
} XL_ReceiveSignalingConfig;
```

C++ Class

```
class XLC_ReceiveSignalingConfig : public
    XLC_ChanRangeMessage {
public:
    unsigned short getStartSpan() const;
    void setStartSpan(unsigned short x);
    UBYTE getStartChannel() const;
    void setStartChannel(UBYTE x);
    unsigned short getEndSpan() const;
    void setEndSpan(unsigned short x);
    UBYTE getEndChannel() const;
```

```

void setEndChannel(UBYTE x);
UBYTE getSignallingType() const;
void setSignallingType(UBYTE x);
UBYTE getSignallingValue() const;
void setSignallingValue(UBYTE x);
UBYTE getTransmissionMode() const;
void setTransmissionMode(UBYTE x);
UBYTE getBitMask() const;
void setBitMask(UBYTE x);
};
    
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0015)	3, 4	Message Type (0x0015)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB)
:	Address Method	10	Checksum
	0x01 Range of AEs		
	Number of AEs to follow		
	AEs		
	0x0D Channel (Starting)		
	0x0D Channel (Ending)		
:	Signaling Type		
	0x01 On-hook		
	0x02 Initial Inseize		
	0x03 Secondary Inseize		
	0x04 Outseize Acknowledgment		
	0x05 Answer		
	0x06 Post Inseize Acknowledgment		
:	Signaling Value		
	Bit 0 A		
	Bit 1 B		
	Bits 2–7 Reserved, must be 0x00		
:	Transmission Mode		
	0x01 A and B Fixed		
	0x02 A Fixed, B Pulsed		
	0x03 A Pulsed, B Fixed		
	0x04 A Pulsed, B Pulsed		
	Pulsed mode timing is determined by ringing on/off timers.		
:	Bit Mask		
	This byte assigns a “Don’t Care” status to bits in the Signaling Value field (for example, 0x01 = consider A bit only, 0x03 = consider A and B bits only).		
:	Checksum		

Example—Inverting E&M Signaling

This example shows the message sequence sent to invert the default E&M Wink Start receive signaling on a channel. The default E&M signaling values are: On-hook: A = 0, B = 0; all other signaling: A = 1, B = 1.

A separate message must be sent to change each signaling type. The examples below show the values which would be sent in each message.

Signaling Type	0x01 (On-hook)
Signaling Value	0x03 (A = 1, B = 1)
Transmission Mode	0x01 (A and B fixed)
Bit Mask	0x03 (A and B)

Signaling Type	0x02 (Initial Inseize)
Signaling Value	0x00 (A = 0, B = 0)
Transmission Mode	0x01 (A and B fixed)
Bit Mask	0x03 (A and B)

Signaling Type	0x03 (Secondary Inseize)
Signaling Value	0x00 (A = 0, B = 0)
Transmission Mode	0x01 (A and B fixed)
Bit Mask	0x03 (A and B)

Signaling Type	0x04 (Outseize Acknowledgment)
Signaling Value	0x00 (A = 0, B = 0)
Transmission Mode	0x01 (A and B fixed)
Bit Mask	0x03 (A and B)

Signaling Type	0x05 (Answer)
Signaling Value	0x00 (A = 0, B = 0)
Transmission Mode	0x01 (A and B fixed)
Bit Mask	0x03 (A and B)

Signaling Type	0x06 (Post Outseize Acknowledgment)
Signaling Value	0x00 (A = 0, B = 0)
Transmission Mode	0x01 (A and B fixed)
Bit Mask	0x03 (A and B)

Record File Modify 0x011F

SwitchKit Name RecordFileModify

Type EXS API and SwitchKit API message

Description **Record File Modify 0x011F**

NOTE: This message applies to the DSP Series 2 card only.

Use this message to modify a current record file session. You can pause, resume, or modify the gain of the recording.

Sent by Host application

Related API Messages *Record File Start, Record File Stop*

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE Action;
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[220];
} XL_RecordFileModify;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[251];
} XL_RecordFileModifyAck;
```

C++ Class

```
class XLC_RecordFileModify : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getAction() const;
```

```
void setAction(UBYTE x);
UBYTE getDataType() const;
void setDataType(UBYTE x);
UBYTE getTLVCount() const;
void setTLVCount(UBYTE x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x)
};
```

C++ Class Response

```
class XLC_RecordFileModifyAck : public
    XLC_OutboundMessage {
public:

    unsigned short getStatus() const
    void setStatus(unsigned short x)
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    ;
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x011F)	3, 4	Message Type (0x011F)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8 :	AIB Address Method 0x00 - Individual AEs	8, 9	Status (MSB, LSB) 0x0001 Invalid TLV Data Software cannot find the TLV Data Buffer. This error can also occur if the Data for a TLV is out of range. 0x0003 Invalid number of TLVs There are no TLVs in the message. 0x0004 Invalid TLV Length The TLV length is different from what is expected. 0x0006 Invalid TLV Unknown TLV 0x000D Mandatory TLVs missing One or more mandatory TLVs are missing Also see Common Response Status Values in the <i>API Reference</i>
	Number of AEs to follow	10	AIB (same as message)
	AEs Use one of the following: 0x0D Channel 0x55 Conference ID 0x45 Child Conference ID and (if recording a conference) 0x01 Slot	11	Checksum
:	Action 0x00 Modify 0x01 Pause 0x02 Resume		
:	Data Type 0x00 TLVs		

Record File Modify 0x011F

:	Number of TLVs to Follow
:	TLVs With Action of Modify 0x05E3 Gain 0x0604 DTMF Clamping/Filtering Enable <u>Positive Voice Detection with RTP</u> If you are using PVD with RTP, the following TLVs are mandatory: 0x0689 - Start/Stop Sending RTP Packets 0x068A - Calender Time Offset for Sending RTP Packets If the Action field is Pause (0x01) or Resume (0x02), no TLVs are used
:	Checksum

Record File Start 0x011E

SwitchKit Name RecordFileStart

Type EXS API and SwitchKit API message

Description **Record File Start 0x011E**

NOTE: This message applies to the DSP Series 2 card only.

Use this message to start a file recording.

You can select the channel and the DSP card, or let the CSP Matrix Series 3 Card choose the DSP Card by leaving the DSP slot set to 0xFF. The CSP Matrix Series 3 Card reports back in the response which DSP Card has been selected.

The Initial Silence Timer specifies how much silence is allowed before the card aborts the recording. The Final Silence Timer specifies how much silence is allowed before the silence triggers the card to stop recording voice.

Any delay between the beginning of conversation and the beginning of recording can result into clipped recorded media. To alleviate this potential problem, Dialogic recommends sending the *Record File Start* message before the conversation begins (before the *Resource Connect* message). Recording sessions involve a signal energy detection threshold, so the silence before conversation will not be recorded.

Sent by Host Application

Related API Message *Record File Modify, Record File Stop*

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_RecordFileStart;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[251];
} XL_RecordFileStartAck;
```

C++ Class

```
class XLC_RecordFileStart : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const; }
.....
```

```

    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getDataType() const;
    void setData(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x)
};

```

C++ Class Response

```

class XLC_RecordFileStartAck : public XLC_OutboundMessage
{
public:

    unsigned short getStatus() const
    void setStatus(unsigned short x)
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    ;
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x011E)	3, 4	Message Type (0x011E)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB)
:	Address Method 0x00 - Individual		0x0001 Invalid TLV Data Software can't find the TLV Data Buffer. This can also occur if the Data for a TLV is out of range.
			0x0003 Invalid number of TLVs There are no TLVs in the message.
			0x0004 Invalid TLV Length The TLV length is different from what is expected.
			0x0006 Invalid TLV Unknown TLV
			0x000D Mandatory TLVs missing One or more mandatory TLVs are missing
			Also see Common Response Status Values in the <i>API Reference</i>
	Number of AEs to follow	10	AIB (same as message)
	AEs You must use one of these AEs: 0x0D Channel* 0x55 Conference ID 0x45 Child Conference ID followed by the 0x01 Slot AE.	11	Checksum
	Notes: If the slot is set to 0xFF, the CSP Matrix Series 3 Card chooses the slot. When recording RTP and there is more than one DSP Series 2 card in your system, you must specify the card you want to use (do not use 0xFF). To record a full duplex voice conversation use two Channel AEs and the Dual Channel Record OptionTLV (0x05ED).		

:	Data Type 0x00 TLVs
:	Number of TLVs to Follow
:	<p>TLVs</p> <p>Mandatory</p> <p>0x05E0 File ID</p> <p>0x05E1 File Format</p> <p>If File ID is 0x00100000 or higher:</p> <p>0x05E2 File Location</p> <p>Optional</p> <p>0x05E3 Gain</p> <p>0x05E6 File Event Descriptor</p> <p>0x05E8 Beep Tone Parameters</p> <p>0x05E9 Initial Silence Timer</p> <p>0x05EA Final Silence Timer</p> <p>0x05EB Maximum Record Timer</p> <p>0x05ED Dual Channel Record Option (Mandatory if you use 2 Channel AEs)</p> <p>0x05FB Silence Threshold</p> <p>0x0604 DTMF Clamping/Filtering Enable</p> <p>0x0615 Append or Replace</p> <p><u>Media Streaming over RTP</u></p> <p>Important: The DSP-2 card supports up to 15 RTP sessions.</p> <p>Mandatory TLVs</p> <p>0x0687 - Enable RTP for Play/Record File</p> <p>0x29FF Media Local End Point Information</p> <p>0x2A00 Media Remote End Point Information</p> <p>0x2A01 Media Per Stream Information</p> <p>0x2A02 Media Per Codec Information</p> <p>0x2A07 Media Port</p> <p>0x2A0E Media Connection Address</p> <p>NOTE: if using RTP, the File ID must be below 0x00100000, so the File Format and File Location TLVs are not required.</p> <p>Optional TLVs</p> <p>0x0688 - RTP Record Mode (to use PVD)</p>
:	Checksum

Record File Stop 0x0120

SwitchKit Name	RecordFileStop
Type	EXS API and SwitchKit API message
Description	<p>Record File Stop 0x0120</p> <p>NOTE: This message applies to the DSP Series 2 card only. Use this message to end a current record file session.</p>
Sent by	Host Application
Related API Messages	<i>Record File Start, Record File Modify</i>

SwitchKit Code C Structure

```
typedef struct {
    UBYTE AddrInfo[253];
} XL_RecordFileStop;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[251];
} XL_RecordFileStopAck;
```

C++ Class

```
class XLC_RecordFileStop : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
};
```

C++ Class Response

```
class XLC_RecordFileStopAck : public XLC_OutboundMessage
{
public:
    unsigned short getStatus() const
```

```
void setStatus(unsigned short x)
const UBYTE *getAddrInfo() const;
UBYTE *getAddrInfo();
void setAddrInfo(UBYTE *x);
XBYTE getConferenceID() const;
void setConferenceID(XBYTE x);
XBYTE getParentConferenceID() const;
void setParentConferenceID(XBYTE x);
XBYTE getChildConferenceID() const;
void setChildConferenceID(XBYTE x);
XBYTE getSpan() const;
void setSpan(XBYTE x);
UBYTE getChannel() const;
void setChannel(UBYTE x);
;
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x0120)	3, 4	Message Type (0x0120)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
:	AIB Address Method 0x00 - Individual AEs	8, 9	Status (MSB, LSB) 0x0001 Invalid TLV Data Software can't find the TLV Data Buffer. This can also occur if the Data for a TLV is out of range. 0x0003 Invalid number of TLVs There are no TLVs in the message. 0x0004 Invalid TLV Length The TLV length is different from what is expected. 0x0006 Invalid TLV Unknown TLV 0x000D Mandatory TLVs missing One or more mandatory TLVs are missing Also see Common Response Status Values.
	Number of AEs to follow	:	AIB Same as message
	AEs Use one of the following: 0x0D Channel 0x55 Conference ID 0x45 Child Conference ID and (if recording a conference) 0x01 Slot	:	Checksum
	Checksum		

RedundantAppPoolsQuery

Type	SwitchKit API message
Description	<p>Use the <i>SK_RedundantAppPoolsQuery</i> message to determine which Redundant Application Pools (RAPs) are defined within the Low-Level Communicator. For example, using this message you can determine which application is the primary one. You cannot modify arguments in this query.</p> <p>The LLC responds to the original query with the message <i>SK_RedundantAppPoolsQueryAck</i>. The response contains a list of all currently defined RAPs within the LLC. The information given in the list can be used to query individual RAPs with <i>SK_RedundantAppQuery</i> or <i>SK_RedundantAppPoolMembersQuery</i>.</p>
Syntax	<p>The data field of this message contains a sequence of null-terminated strings, where each string is the name of a RAP. See the following example:</p> <pre>Data[RAPa\ORAPb\ORAPc\0]</pre> <p>This indicates that there are three RAPs, named “RAPa”, “RAPb”, and “RAPc”.</p>
Sent by	Monitoring application or Converged Services Administrator (CSA)
C Structure	<pre>typedef struct { } SK_RedundantAppPoolsQuery;</pre>
C Structure Response	<pre>typedef struct { int Status; int NumRAPs; unsigned short DataSize; UBYTE Data[243]; } SK_RedundantAppPoolsQueryAck;</pre>
C++ Class	<pre>class SKC_RedundantAppPoolsQuery : public SKC_ToolkitMessage { public: };</pre>
C++ Class Response	<pre>class SKC_RedundantAppPoolsQueryAck : public SKC_ToolkitAck { public: int getStatus() const; void setStatus(int x);</pre>

RedundantAppPoolsQuery

```
int getNumRAPs() const;
void setNumRAPs(int x) ;
unsigned short getDataSize() const;
void setDataSize(unsigned short x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};
```

Redundant App Pool Members Query

Type SwitchKit API message

Description Use this message to determine which applications are members of a specific Redundant Application Pool (RAP).

The LLC responds to the original query with the message *SK_RedundantAppPoolMembersQueryAck*. The response includes a list of application IDs that are currently members of a RAP. The response also indicates which application is the primary application in the pool.

Sent by Monitoring application or Converged Services Administrator (CSA).

Arguments The following table shows the arguments you can modify:

Argument	Description
RedundantAppPoolID	The RedundantAppPoolID is a string that uniquely identifies the class of application wishing to be treated as redundant applications.

Notes: RAP IDs are case sensitive!

C Structure

```
typedef struct {
    char RedundantAppPoolID[32];
} SK_RedundantAppPoolMembersQuery;
```

C Structure Response

```
typedef struct {
    int Status;
    char RedundantAppPoolID[32];
    int PrimaryAppName;
    int NumAppNames;
    int AppNames[10];
} SK_RedundantAppPoolMembersQueryAck;
```

C++ Class

```
class SKC_RedundantAppPoolMembersQuery : public
    SKC_ToolkitMessage {
public:
    const char *getRedundantAppPoolID() const;
    void setRedundantAppPoolID(const char *x);
};
```

C++ Class Response

```
class SKC_RedundantAppPoolMembersQueryAck : public
    SKC_ToolkitAck {
public:
    int getStatus() const;
```

Redundant App Pool Members Query

```
void setStatus(int x);  
const char *getRedundantAppPoolID() const;  
void setRedundantAppPoolID(const char *x);  
int getPrimaryAppName() const;  
void setPrimaryAppName(int x);  
int getNumAppNames() const;  
void setNumAppNames(int x);  
const int *getAppNames() const;  
void setAppNames(const int *x);  
};
```

RedundantAppQuery

Type	SwitchKit API message
Description	Use the <i>SK_RedundantAppQuery</i> message to determine the specifications of a member application within a specific Redundant Application Pool (RAP). Since an application can be a member of several RAPs, <i>SK_RedundantAppQuery</i> needs to be sent for each RAP of which an application is a member.
Sent by	Monitoring application or Converged Services Administrator (CSA)
Arguments	The following table shows the user modifiable arguments:

Argument	Description
AppName	AppName is the application name of the requesting application. The name can be obtained by calling <code>sk_getConnectionName()</code> .
RedundantAppPoolID	The RedundantAppPoolID is a string that uniquely identifies the class of application wishing to be treated as redundant applications.

Notes: RAP IDs are case sensitive!

C Structure

```
typedef struct {
    char RedundantAppPoolID[32];
    int AppName;
} SK_RedundantAppQuery;
```

C Structure Response

```
typedef struct {
    int Status;
    char RedundantAppPoolID[32];
    int AppName;
    int RedundantAppPriority;
    int IsPrimary;
    unsigned short DataSize;
    UBYTE Data[203];
} SK_RedundantAppQueryAck;
```

C++ Class

```
class SKC_RedundantAppQuery : public SKC_ToolkitMessage {
public:
    const char *getRedundantAppPoolID() const;
    void setRedundantAppPoolID(const char *x);
    int getAppName() const;
```

```
void setAppName(int x);  
};
```

C++ Class Response

```
class SKC_RedundantAppQueryAck : public SKC_ToolkitAck {  
public:  
    int getStatus() const;  
    void setStatus(int x);  
    const char *getRedundantAppPoolID() const;  
    void setRedundantAppPoolID(const char *x);  
    int getAppName() const;  
    void setAppName(int x);  
    int getRedundantAppPriority() const;  
    void setRedundantAppPriority(int x);  
    int getIsPrimary() const;  
    void setIsPrimary(int x);  
    unsigned short getDataSize() const;  
    void setDataSize(unsigned short x);  
    const UBYTE *getData() const;  
    UBYTE *getData();  
    void setData(UBYTE *x);  
};
```

RedundantAppStatusMsg

Type SwitchKit API message

Description The LLC sends *SK_RedundantAppStatusMsg* to notify members of a Redundant Application Pool (RAP) about any changes to the state of the RAP.

This message is sent when an application first registers for membership and when the status of a member application changes. Status changes include the following:

- An application registers for membership
- An application sends the *SK_ReselectPrimaryApp* message
- An application deregisters from membership in the RAP
- An application changes to primary

Sent by LLC

Arguments You can modify the following arguments:

Argument	Description
AppName	AppName is the application name of the requesting application. The name can be obtained by calling <i>sk_getConnectionName()</i> .
RedundantAppPoolID	The RedundantAppPoolID is a string that uniquely identifies the class of application wishing to be treated as redundant applications. Notes: RAP IDs are case sensitive.

Argument	Description
RedundancyStatus	<p>Values defined for this argument are:</p> <ul style="list-style-type: none"> • SK_RED_STATUS_REMOVED (0) - The specified application has been removed from the redundant application. • SK_RED_STATUS_PRIMARY (1) - The specified application is the primary app for the redundant app group. • SK_RED_STATUS_SECONDARY (2) - The specified application is a secondary application for the redundant application group. • SK_RED_STATUS_NO_PRIMARY (3) -There is no primary application for the redundant app group. • SK_RED_STATUS_MONITOR (4) - The specified application is a monitoring application of the redundant application group.

C Structure

```
typedef struct {
    int AppName;
    char RedundantAppPoolID[32];
    UBYTE RedundancyStatus;
} SK_RedundantAppStatusMsg;
```

C++ Class

```
class SKC_RedundantAppStatusMsg : public
    SKC_ToolkitInbound {
public:
    int getAppName() const;
    void setAppName(int x);
    const char *getRedundantAppPoolID() const;
    void setRedundantAppPoolID(const char *x);
    UBYTE getRedundancyStatus() const;
    void setRedundancyStatus(UBYTE x);
};
```

Redundant LLC Register

Type	SwitchKit API message
Description	<p>This message is used internally between the primary LLC and the redundant LLC.</p> <p>This message does not have a response.</p>
Sent by	LLC
C Structure	<pre>typedef struct { } <i>SK_RedundantLLCRegister</i>;</pre>
C++ Class	<pre>class <i>SKC_RedundantLLCRegister</i> : public SKC_AdminMessage { public: };</pre>

RegisterAsRedundantApp

Type SwitchKit API message

Description The *SK_RegisterAsRedundantApp* message is used by the function `sk_registerAsRedundant()`.

If an application needs to register for membership into a Redundant Application Pool (RAP), the application must call the function `sk_registerAsRedundant()`. The function then uses the *SK_RegisterAsRedundantApp* message to register the application for membership into a RAP.

To remove an application from a pool, the application calls the function `sk_deregisterAsRedundant()`, which then also uses the message to take a member out of the pool.

A *SK_RegisterAsRedundantAppAck* message is an acknowledgment by the LLC, indicating the completion of the original request.

Sent by Function `sk_registerAsRedundant()`

Arguments You can modify the following arguments:

Argument	Description
AppName	AppName is the application name of the requesting application. The name can be obtained by calling <code>sk_getConnectionName()</code> .
RedundantAppPoolID	The RedundantAppPoolID is a string that uniquely identifies the class of application wishing to be treated as redundant applications. Notes: RAP IDs are case sensitive.

Argument	Description
RedundantAppPriority	<p>RedundantAppPriority is a value indicating the current priority of the application. The higher the value, the more likely it is that the LLC will select the application as primary. Predefined values are:</p> <p>SK_RED_APP_PRI_MONITOR (0) Used by an application wishing to monitor the activity of this RedundantAppClass. An application selecting this priority cannot be considered either primary or secondary.</p> <p>SK_RED_APP_PRI_REMOVED (-1) Used when an application needs to be removed from an RedundantAppClass for which it has previously registered.</p>
DataSize	Size of the data
Data	Data bytes

C Structure

```
typedef struct {
    int AppName;
    char RedundantAppPoolID[32];
    int RedundantAppPriority;
    unsigned short InitiateSwitchover;
    unsigned short DataSize;
    UBYTE Data[211];
} SK_RegisterAsRedundantApp;
```

C Structure Response

```
typedef struct {
    int Status;
    int AppName;
    char RedundantAppPoolID[32];
    int RedundantAppPriority;
    unsigned short InitiateSwitchover;
} SK_RegisterAsRedundantAppAck;
```

C++ Class

```
class SKC_RegisterAsRedundantApp : public
    SKC_ToolkitMessage {
public:
    int getAppName() const;
    void setAppName(int x);
    const char *getRedundantAppPoolID() const;
    void setRedundantAppPoolID(const char *x);
    int getRedundantAppPriority() const;
    void setRedundantAppPriority(int x);
    unsigned short getInitiateSwitchover() const;
    void setInitiateSwitchover(unsigned short x);
    unsigned short getDataSize() const;
    void setDataSize(unsigned short x);
    const UBYTE *getData() const;
    UBYTE *getData();
```

RegisterAsRedundantApp

```
void setData(UBYTE *x);  
};
```

C++ Class Response

```
class SKC_RegisterAsRedundantAppAck : public  
    SKC_ToolkitAck {  
public:  
    int getStatus() const;  
    void setStatus(int x);  
    int getAppName() const;  
    void setAppName(int x);  
    const char *getRedundantAppPoolID() const;  
    void setRedundantAppPoolID(const char *x);  
    int getRedundantAppPriority() const;  
    void setRedundantAppPriority(int x);  
    unsigned short getInitiateSwitchover() const;  
    void setInitiateSwitchover(unsigned short x)  
};
```

Register Virtual Name

Type	SwitchKit API message
Description	This message is used internally between LLC and RLLC.
Sent by	LLC
C Structure	<pre>typedef struct { } SK_RegisterVirtualName;</pre>
C++ Class	<pre>class SKC_RegisterVirtualName : public SKC_AdminMessage { public: };</pre>

Release Channel 0x0008

SwitchKit Name	ReleaseChannel
Type	EXS API and SwitchKit API message
Description	<p>Release Channel 0x0008</p> <p>Use this message to initiate the release of a single channel or a connected pair of channels. The host receives a <i>Channel Released</i> message when the distant-end releases.</p> <p>To release only one channel, enter the channel in both the <i>Logical Span ID</i> and <i>Channel</i> fields. The other connected channel is parked.</p>
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short SpanA;
    UBYTE ChannelA;
    unsigned short SpanB;
    UBYTE ChannelB;
} XL_ReleaseChannel;
```

C++ Class

```
class XLC_ReleaseChannel : public XLC_OutboundMessage {
public:
    unsigned short getSpanA() const;
    void setSpanA(unsigned short x);
    UBYTE getChannelA() const;
    void setChannelA(UBYTE x);
    unsigned short getSpanB() const;
    void setSpanB(unsigned short x);
    UBYTE getChannelB() const;
    void setChannelB(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0008)	3, 4	Message Type (0x0008)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB) The LSB is the Call State when the Nack value is 0x1D. Otherwise the LSB is 00.
:	Address Method		
	Number of AEs to follow	10	Checksum
	AEs		
	0x0D Channel A or 0x0D Channel B Specify either Channel A or Channel B whichever channel did not initiate the release. You must specify the channel twice.		
:	Checksum		

Response Status

Call State	CSP Call Control CH Values
Out-of-Service	0x00 Out-of-Service
Idle	0x03 In Service Idle
Connect Wait	0x04 Incoming Call, Wait for L5
Book Wait	0x06 Wait For CSA in Incoming Call Wait for Host State
	0x01 Incoming Call
Answer Wait 2	0x0E Incoming Call Alerted
Connect	0x07 Answered
	0x02 Wait For CSA in Incoming Call State
	0x0C Wait for CSA in Incoming Alerted State
	0x0D Wait for CSA in Incoming in Answered State
Clear Wait	0x09 L3 Clear Wait
	0x0F L3 Clear Wait
Busied Out	0x0A Busied out
	0x10 Wait for CSA in Outgoing Cut-thru
Release With Data Wait	0x15 L5 Release Wait
Outseize Wait	0x05 Outgoing Call, L3 Outseize Wait
Parked	0x13 Outgoing Call Alerted
	0x11 Wait for CSA in Outgoing Alerted State
Outseized	0x0B Outgoing Call Cut-thru
	0x12 L4 Clear ACK Wait, L3 Clear Received
Release Wait	0x08 L4 Clear ACK Wait, L3 Disconnect Received
	0x14 L4 Clear ACK Wait, L5 Clear Received
	0x16 Incoming Call, L4 Clear ACK Wait (Park Processing)
	0x17 Incoming Call Alerted, L4 Clear ACK Wait (Park Processing)
	0x18 Answered, L4 Clear ACK Wait (Park Processing)
Call State	0x19 Outgoing Alerted, L4 Clear ACK Wait (Park Processing)
	CSP Call Control CH Values
	0x1A Outgoing Cut-thru, L4 Clear ACK Wait (Park Processing)
	0x1B Wait for CSA for Internal Routing

Release Channel With Data 0x0036

SwitchKit Name	ReleaseWithData
Type	EXS API and SwitchKit API message
Description	<p>Release Channel With Data 0x0036</p> <p>Use this message to release a channel with network-specific information elements that supply the network with host-specific information elements. This message is typically used with ISDN controlled channels.</p>
Sent by	Host

SwitchKit Name **C Structure**

```
typedef struct {
    unsigned short SpanA;
    UBYTE ChannelA;
    unsigned short SpanB;
    UBYTE ChannelB;
    UBYTE ReleaseDataType;
    UBYTE Data[222];
} XL_ReleaseWithData;
```

C++ Class

```
class XLC_ReleaseWithData : public XLC_OutboundMessage {
public:
    unsigned short getSpanA() const;
    void setSpanA(unsigned short x);
    UBYTE getChannelA() const;
    void setChannelA(UBYTE x);
    unsigned short getSpanB() const;
    void setSpanB(unsigned short x);
    UBYTE getChannelB() const;
    void setChannelB(UBYTE x);
    UBYTE getReleaseDataType() const;
    void setReleaseDataType(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0036)	3, 4	Message Type (0x0036)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB)
:	Address Method 0x00 - Individual AEs		The LSB is the Call State when the Nack value is 0x1D. Otherwise the LSB is 00.
	Number of AEs to follow	10	Checksum
	AE 0x0D Channel A or 0x0D Channel B Specify either Channel A or Channel B - whichever channel did not initiate the release. You must specify the channel twice. With SS7 calls the host receives a Channel Release Request 0x37 message to identify the channel to release.		
:	Release Data Type 0x01 Reserved 0x03 ISDN ICB Formatted IE and Raw IE or DASS 2/DPNSS Raw IE Data 0x04 SS7 ICB 0x33 NPDI Universal ICB		
	Number of ICBs to follow (Ignore this field if no ICBs in message.)		

Release Channel With Data 0x0036

:	0x02 Data 0x10 ISDN Formatted IEs 0x11 ISDN Raw IEs 0x12 SS7 Parameters 0x15 DASS2/DPNSS Raw Data 0x1C SS7 TUP Formatted Fields 0x23 BT IUP Parameters 0x25 ISDN Segmented Message 0x03 Extended 0x0033 NPDI Universal ICB
	Refer to <i>Interworking TLVs (4-4)</i> if you have enabled interworking from the <i>VoIP Protocol Configure 0x00EE</i> message.
:	Checksum

ReleaseCacheChannels

Type	SwitchKit API message
Description	This message is used by the function <code>sk_returnChannel()</code> to make channel available.
Sent by	Function <code>sk_returnChannel()</code>
C Structure	<pre>typedef struct { } SK_ReleaseCacheChannels;</pre>
C++ Class	<pre>class SKC_ReleaseCacheChannels : public SKC_AdminMessage { public: };</pre>

Remove Channels From Group

Type	SwitchKit API message
Description	Use this message to remove channels from an existing channel group.
Sent by	SwitchManager or Application
Configuration	<i>RemoveChannelsFromGroup</i> (ChannelGroup = quoted string, range = startSpan:startChannel - endSpan:endChannel);
C Structure	<pre>typedef struct { BaseFields Base; unsigned short StartSpan; unsigned short EndSpan; UBYTE StartChannel; UBYTE EndChannel; char ChannelGroup[45]; UBYTE reserved68[16]; } <i>SK_RemoveChannelsFromGroup</i>;</pre>
C++ Class	<pre>class <i>SKC_RemoveChannelsFromGroup</i>: public SKC_AdminMessage { public: SKC_RemoveChannelsFromGroup(int sz = 0); ~SKC_RemoveChannelsFromGroup(); SK_DECLARE_CLASS(SKC_RemoveChannelsFromGroup, SKC_Admin Message) virtual MsgStruct *getStructPtr(); virtual const MsgStruct *getStructPtr() const; virtual int getTag() const; unsigned short getStartSpan() const; void setStartSpan(unsigned short x); unsigned short getEndSpan() const; void setEndSpan(unsigned short x); UBYTE getStartChannel() const; void setStartChannel(UBYTE x); UBYTE getEndChannel() const; void setEndChannel(UBYTE x); const char *getChannelGroup() const; void setChannelGroup(const char *x); };</pre>

Remove Channels From Group

Example The following is an example of *RemoveChannelsFromGroup* for Channel Group “all”, with a range of span 0/channel 2 - span 1/channel 3.

```
RemoveChannelsFromGroup(  
  ChannelGroup="all",  
  Range=0:2-1:3);
```

RemoveLLCRoutingID

Type SwitchKit API message

Overview This message will remove an individual RoutingID from the LLC but leave the connection to the card and all other RoutingID's established even after the last RoutingID is removed. However, if an attempt is made to remove the last RoutingID before removing the connection, the LLC will log a message stating that the devices have been orphaned and will then remove the RoutingID. At this point the LLC maintains communications with the card pair but the socket pair is reported as being down so that applications will not attempt to send/receive messages to it. After a pair of devices have been orphaned (all RoutingID's have been removed), routing and communications for an application can be re-established without establishing a new connection by configuring a new RoutingID for that card IP pair using the *AddLLCCard* message.

Sent by Application

C Structure

```
typedef struct {
    BaseFields Base;
    int RoutingID;
    unsigned short CardType;
    UBYTE reserved23[4];
} SK_RemoveLLCRoutingID;
```

C Structure Response

```
typedef struct {
    BaseFields Base;
    int Status;
    int RoutingID;
    UBYTE Reserved1;
    UBYTE reserved26[5];
} SK_RemoveLLCRoutingIDAck;
```

C++ Class

```
class SKC_RemoveLLCRoutingID : public
    SKC_ToolkitMessage {
public:
    SKC_RemoveLLCRoutingID(int sz = 0);
    ~SKC_RemoveLLCRoutingID();

    SK_DECLARE_CLASS(SKC_RemoveLLCRoutingID, SKC_ToolkitMes
    sage)

    virtual MsgStruct *getStructPtr();
```

RemoveLLCRoutingID

```
virtual const MsgStruct *getStructPtr() const;
virtual int getTag() const;

int getRoutingID() const;
void setRoutingID(int x);
unsigned short getCardType() const;
void setCardType(unsigned short x);
```

C++ Class Response

```
class SKC_RemoveLLCRoutingIDAck : public
    SKC_ToolkitAck {
public:
    SKC_RemoveLLCRoutingIDAck(int sz = 0);
    ~SKC_RemoveLLCRoutingIDAck();

    SK_DECLARE_CLASS(SKC_RemoveLLCRoutingIDAck, SKC_Toolkit
        Ack)

    virtual MsgStruct *getStructPtr();
    virtual const MsgStruct *getStructPtr() const;
    virtual int getTag() const;

    int getStatus() const;
    void setStatus(int x);
    int getRoutingID() const;
    void setRoutingID(int x);
    UBYTE getReserved1() const;
    void setReserved1(UBYTE x);
```

RemoveLLCRoutingIDAck

The LLC will acknowledge the RemoveLLCRoutingIDAck with a RemoveLLCRoutingAck message.

```
RemoveLLCRoutingIDAck (Status=<**Response Value>
RoutingID=      <Unique ID for Card Pair>,
Reserved1=      <Not currently used>);
```

Status Field

Possible Response Values found in the Status field of an RemoveLLCRoutingIDAck Message.

**Response Values	Action
SK_SUCCESS	The RoutingID was successfully removed.
SK_NOT_PRESENT	The RoutingID was not found.
SK_NO_MESSAGE	Invalid message parameter found while processing request.

RoutingID Macros

To avoid duplicate values, the routing ID for each type of sub-component card must be derived using that cards corresponding get_RoutingID macro. Macros will be defined within SK_API.h so that they are globally accessible.

Macros available for deriving RoutingID's

CardType	Function
SK_SS7TCAP_BOARD (0x03)	getSS7TCAP_RoutingID(int stackid(0-255));

RequestChannelAck

Type SwitchKit API message

Description The LLC sends this message as a response to a previously sent **sk_requestOutseizedChannel()** or **sk_requestChannel()**. It notifies the application that a channel has been assigned to it, which must later be returned with **sk_returnChannel()**. It is important to note that even in the case of a **SK_OUTSEIZE_NACK**, a channel has been assigned to the application which must be returned.

To check the return status, you should first check **SKStatus**. Only if **SKStatus** contains **SK_OUTSEIZE_NACK** should **XLStatus** be examined. Otherwise, **XLStatus** will contain invalid data.

SKStatus will contain one of the following values:

- **OK** - the channel referenced by “Span/Channel” was successfully outseized. Span/Channel is considered allocated to this application, and must be returned through **sk_returnChannel** when the application is done with it.
- **SK_OUTSEIZE_NACK** - the channel Span/Channel received an **XLStatus** nack from the Outseize attempt. This will only be returned after the number of retries specified in the **sk_requestOutseizedChannel()** have all been tried and failed. The last Span, Channel whose outseize failed is considered allocated to this application, and must be returned through **sk_returnChannel()** when the application is done with it.
- **SK_NO_CHANNELS** - none of the channels in the requested group were available. The channels might be unavailable because they’re in use, out of service, or in yellow or red alarm. In this case, Span/Channel will contain invalid data, as no channel has been assigned to the application.

Sent by LLC

C Structure

```
typedef struct {
    int SKStatus;
    unsigned short XLStatus;
    int Span;
    int Channel;
} SK_RequestChannelAck;
```

C++ Class `class SKC_RequestChannelAck : public SKC_ToolkitAck {`

RequestChannelAck

```
public:  
    int getSKStatus() const;  
    void setSKStatus(int x);  
    unsigned short getXLStatus() const;  
    void setXLStatus(unsigned short x);  
    int getSpan() const;  
    void setSpan(int x);  
    int getChannel() const;  
    void setChannel(int x);  
};
```

RequestChannelMsg

Type	SwitchKit API message
Description	This message is used by the function <i>sk_requestOutseizedChannel()</i> when an application calls for outseize channels. For more information about <i>sk_requestOutseizedChannel()</i> see the <i>SwitchKit Programmer's Guide</i> .
Sent by	Function <i>sk_requestOutseizedChannel()</i>
C Structure	<pre>typedef struct { } SK_RequestChannelMsg;</pre>
C++ Class	<pre>class SKC_RequestChannelMsg : public SKC_AdminMessage { public: };</pre>

Request For Service 0x0040

SwitchKit Name	Request For Service
Type	EXS API and SwitchKit API message
Description	<p>Request For Service 0x0040</p> <p>This message is sent to the host when an incoming call is detected. Valid host responses to this message include the following:</p> <p><i>Connect</i></p> <p><i>Connect With Pad</i></p> <p><i>Connect One-Way Forced</i></p> <p><i>Connect With Data</i></p> <p><i>Connect Wait</i></p> <p><i>Connect to Conference</i></p> <p><i>Connect One-Way to Conference</i></p> <p><i>Park Channel</i></p>
Sent by	CSP
Resent	This message is resent once after five seconds.
SwitchKit Code	<p>C Structure</p> <pre>typedef struct { unsigned short Span; UBYTE Channel; UBYTE ResendFlag; } XL_RequestForService;</pre> <p>C++ Class</p> <pre>class XLC_RequestForService : public XLC_OneChannelMessage { public: unsigned short getSpan() const; void setSpan(unsigned short x); UBYTE getChannel() const; void setChannel(UBYTE x); UBYTE getResendFlag() const; void setResendFlag(UBYTE x); };</pre>

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type, MSB (0x00)	3, 4	Message Type, MSB (0x00)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8	AIB
:	Address Method	:	Checksum
	0x00 - Individual AEs		
	Number of AEs to follow		
	AE		
	0x0D Channel		
:	Resend Flag		
	0x00 <i>Request for Service</i> is not a resend		
	0x01 <i>Request for Service</i> is a resend		
:	Checksum		

Request For Service With Data 0x002D

SwitchKit Name	RFSWithData
Type	EXS API and SwitchKit API message
Description	<p>Request For Service With Data 0x002D</p> <p>This message is sent to the host when an incoming call is detected on the specified channel and the specified channel was previously configured to “report incoming call with digits.”</p> <p>For SS7, the system uses this message as a response to an IAM message from the network. By default, BT IUP parameters are reported to the host in a BT IUP parameters ICB. The host can modify this to have address data sent as BCD-encoded digits.</p> <p>SIP endpoints report incoming calls from the SIP side using the <i>Request for Service with Data</i> message. The data type Universal Data (0x33) is for SIP calls. The UPDF parameters supported are the same as those supported in the <i>Route Control</i> message.</p> <p>H.323 endpoints report incoming calls from the H.323 side using the <i>Request for Service with Data</i> message. The data type Universal Data (0x33) is for H.323 calls. The UPDF parameters supported are the same as those supported in the <i>Route Control</i> message.</p>
Sent by	CSP
Resent	This message is resent once after five seconds. It includes a field to indicate it has been resent.
Example Message (Socket Log Output)	<p>In the example message below, phone 333@10.10.65.37 is calling phone 666. Phone 333 is registered with the CSP, which is at IP address 10.10.65.20.</p> <pre>00 a3 00 2d 00 03 ff 00 01 0d 03 00 01 05 00 33 01 03 00 33 00 8f 00 0f 27 4e 00 02 00 05 27 7e 00 03 08 00 00 29 19 00 04 36 36 36 00 29 1b 00 0c 31 30 2e 31 30 2e 36 35 2e 32 30 00 29 23 00 04 33 33 33 00 29 25 00 0c 31 30 2e 31 30 2e 36 35 2e 32 30 00 27 18 00 07 02 00 00 00 03 33 30 27 17 00 05 02 00 03 66 60 27 94 00 04 0a 0a 41 25 27 95 00 04 00 00 4c dc 27 b0 00 02 00 02 27 b1 00 02 00 01 29 2d 00 04 33 33 33 00 29 2f 00 0c 31 30 2e 31 30 2e 36 35 2e 33 37 00 29 30 00 04 00 00 13 c4</pre>

SwitchKit Code C Structure

```
typedef struct {
    unsigned short Span;
    UBYTE Channel;
    UBYTE ResendFlag;
    UBYTE AddressDataType;
    UBYTE Data[221];
} XL_RFSWithData;
```

C++ Class

```
class XL_C_RFSWithData : public XLC_OneChannelMessage {
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getResendFlag() const;
    void setResendFlag(UBYTE x);
    UBYTE getAddressDataType() const;
    void setAddressDataType(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x002D)	3, 4	Message Type (0x002D)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8	AIB
:	Address Method	:	Same as message.
	0x00 - Individual AEs		Checksum
	Number of AEs to follow		
	AEs		
	0x0D Channel		
:	Resend Flag		
	0x00 <i>Request for Service With Data</i> is not a resend		
	0x01 <i>Request for Service With Data</i> is a resend		

Request For Service With Data 0x002D

:	<p>Address Data Type</p> <p>0x01 BCD-Encoded Stages (not an ICB: see BCD Encoded Stages data below table)</p> <p>0x03 ISDN ICB</p> <p>0x04 Q.931 Received Frame</p> <p>0x05 SS7 ICB</p> <p>0x06 DASS2/DPNSS</p> <p>0x07 BT IUP</p> <p>0x33 NPDI Universal ICB</p>
:	Number of ICBs (or BCD Stages) to follow.
	<p>ICBs</p> <p>For Address Data Type 0x03, the following ICBs can be used:</p> <p>0x10 ISDN Formatted IEs</p> <p>0x11 ISDN Raw IEs</p> <p>0x25 ISDN Segmented Message</p> <p>For Address Data Type 0x05, the following ICBs can be used:</p> <p>0x12 SS7 Parameters</p> <p>0x1C SS7 TUP Formatted Fields</p> <p>0x66 SS7 Address Information</p> <p>For Address Data Type 0x06, the following ICB can be used:</p> <p>0x15 DASS2/DPNSS Raw Data</p> <p>For Address Data Type 0x07, the following ICB can be used:</p> <p>0x23 BT IUP Parameters</p> <p>For Address Data Type 0x33, the following ICB can be used:</p> <p>0x0033 NPDI Universal ICB</p>
:	Data[0] (See information below table)
	Refer to <i>Interworking TLVs (4-4)</i> if you have enabled interworking from the <i>VoIP Protocol Configure 0x00EE</i> message.
:	Checksum

Data 0x01 BCD-Encoded Stages

Data[0] Stage Count (X)

Data[1] Stage 1 Status

Data[2] Stage 1 String Count

Data[3] Stage 1: String 1Digit Count

Data[4] Stage 1: String 1First BCD-Encoded digit pair

Data[5] :

Data[n] Stage 1: String 1 Last BCD-Encoded digit pair

Data[...]₁ Stage 1: String 2 Digit Count

Data[...]₂ Stage 1: String 2 First BCD-Encoded digit pair

Data[...]₃ :

Data[...]_x Stage 1: String x Last BCD-Encoded digit pair

: :

Data[...]Stage X Status
Data[...]Stage X String Count
Data[...]Stage X: String 1Digit Count
Data[...]Stage X: String 1First BCD-Encoded digit pair
Data[...] :
Data[N+1]Stage X: String x Last BCD-Encoded digit pair
Stage Status

The system reports the stage status for each of the five stages. Timers can be configured.

Valid entries for this field are as follows:

- 0x10 Positive Acknowledgment – Digits received
- 0x80 Partial Dial Condition – Receive Inter-digit Duration timer expired
- 0x81 Permanent Signal Condition – Receive 1st Digit Detection Timer expired
- 0x91 Inpulse Stage Not Collected – Stage not collected; no further stage information will be reported.
- 0x92 Digit Complete Timeout

NOTES:

- 1 Each stage may contain up to two strings.
- 2 Valid responses to this message may also be *Connect*, *Connect With Pad*, *Connect One-Way Forced*, *Connect With Data*, *Connect Wait*, *Connect To Conference*, *Connect One-Way To Conference*, and *Park Channel*.

0x04 Q.931 Received Frame

This data is allowed for the ISDN interface only and will be the exact data received from the distant end (see ITU Q.931).

Data[0] Protocol Discriminator
Data[1] Call Reference Length
Data[2] :

RequestStandbyPoll

Type	SwitchKit API message
Description	Use this message to initiate the sending of a poll message from a node's standby CSP Matrix Series 3 Card. Important! Polls cannot be requested from SS7 cards.
Sent by	Application
C Structure	<pre>typedef struct { } SK_RequestStandbyPoll;</pre>
C++ Class	<pre>class SKC_RequestStandbyPoll : public SKC_AdminMessage { public: };</pre>

Reselect Primary App

Type SwitchKit API message

Description Use the *SK_ReselectPrimaryApp* message to indicate to the LLC that the primary application should be selected for a specific Redundant Application Pool (RAP). By default, the first application to connect to the LLC is the primary application. You can change which application is designated as primary, using the appropriate variable. After receiving this message, the LLC looks at all members of a RAP and selects a new primary application for the RAP. The Flag field dictates the behavior of this command as described in the arguments table further on this page.

If the LLC selects a new primary application, all members of the specific RAP are notified about the change in status, including the indication.

The *SK_ReselectPrimaryAppAck* is the response to the initial *SK_ReselectPrimaryApp* message, and it indicates the success or failure of the reselect.

Sent by Application

Arguments The following table shows the arguments that you can modify:

Argument	Description
RedundantAppPoolID	<p>The RedundantAppPoolID is a string that uniquely identifies the class of application wishing to be treated as redundant applications.</p> <p>Notes: RAP IDs are case sensitive!</p>
Flag	<p>The Flag indicates the reselect behavior.</p> <ul style="list-style-type: none"> • SK_RED_RESELECT_DEFAULT (0) - Execute primary selection algorithm using priority. The application with the highest priority is selected as primary. No change is made if the value of the current primary is equal to or higher than of all other applications. • SK_RED_RESELCT_PRIMARY (1) - The requesting application should be made primary. The Application must be a member of the RAP and cannot be a monitor of that RAP. • SK_RED_RESELCT_SECONDARY (2) - the requesting application should be made secondary. If the application is already secondary, this request has no impact on the RAP.

Status	The Status is the result of the original request. A status of zero (0) indicates a success. Any other value indicates an error.
--------	---

C Structure typedef struct {
 char RedundantAppPoolID[32];
 int Flag;
 UBYTE reserved53[217]
 } **SK_ReselectPrimaryApp**;

C Structure Response typedef struct {
 int Status;
 UBYTE reserved21[4];
 char RedundantAppPoolID[32];
 UBYTE reserved57[213]
 } **SK_ReselectPrimaryAppAck**;

C++ Class class **SKC_ReselectPrimaryApp** : public SKC_ToolkitMessage {
 public:
 const char *getRedundantAppPoolID() const;
 void setRedundantAppPoolID(const char *x);
 int getFlag() const;
 void setFlag(int x);
 };

C++ Class Response class **SKC_ReselectPrimaryAppAck** : public SKC_ToolkitAck {
 public:
 int getStatus() const;
 void setStatus(int x);
 const char *getRedundantAppPoolID() const;
 void setRedundantAppPoolID(const char *x);
 };

Reset Configuration 0x000B

SwitchKit Name	ResetConfig
Type	API and SwitchKit API message
Description	<p>Reset Configuration 0x000B</p> <p>This message resets a card's configuration to the default values by corrupting the battery-backed configuration and initiating a hardware reset on the local processor. To reset the configuration on a CSP Matrix Series 3 Card, all cards must be specified and the force flag must be set.</p> <p>This message is useful for completely reinitializing the CSP or any component in the CSP. Generally, it should not be used on a live system.</p> <p>For SwitchKit, if you have included <i>ResetConfig</i> messages in your configuration file, the <i>ConfigStage</i> must be set to 10 for a reset to be performed. If the <i>ConfigStage</i> is anything other than 10, the reset is ignored.</p>
Sent by	Host

SwitchKit Code Configuration

```
ResetConfig (
    Node = integer,
    Slot = integer,
    ForcedFlag = integer);
```

C Structure

```
typedef struct {
    UBYTE Slot;
    UBYTE ForcedFlag;
} XL_ResetConfig;
```

C++ Class

```
class XLC_ResetConfig : public XLC_OutboundMessage {
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getForcedFlag() const;
    void setForcedFlag(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x000B)	3, 4	Message Type (0x000B)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB
:	Address Method	10	Checksum
	0x00 - Individual AEs		
	Number of AEs to follow		
	AEs		
	0x01 Slot		
	The location of the line card to be reset (0xFF = all cards in the system)		
:	Force Flag		
	0xFF: Card configuration will be reset even if it has spans in service. All cards in the system will have their configuration reset.		
	If this flag is not set, messages sent to a card with spans in service fail, and the host receives a Response Status of 0x72 (Span In Service).		
:	Checksum		

Notes:

1. When resetting a line card's configuration, the host must wait for a *Card Status Report* message on the specified line card before considering its configuration reset.
2. Switchover will occur if all cards in the system are specified and there is a standby CSP Matrix Series 3 Card present.
3. The CSP Matrix Series 3 Card that gains control of the system after you send the *Reset Configuration* message becomes the active CSP Matrix Series 3 Card. This CSP Matrix Series 3 Card may not necessarily be the one that was active before you sent the message.

Reset IP Signaling Series 3 Card 0x0107

Type	EXS API message
Description	Reset IP Signaling Series 3 Card 0x0107 This message resets the IP Signaling Series 3 card before and/or after the IP Signaling Series 3 Card Interface is established. This message is sent directly to the IP Signaling Series 3 card and uses the extended logical node ID format. You can also reset the IP Signaling Series 3 card using the <i>Reset Configuration 0x000B</i> message once the IP Signaling Series 3 Card Interface is established.
Sent by	Host
Example Message	The follow sample resets the IP Signaling Series 3 card when the IP Signaling Series 3 card ID or Matrix ID is not assigned.

```
00 11 01 07 00 00 fe 00 02 52 02 ff ff 53 04 00 20 ff ff
```

SwitchKit Code **C Structure**

```
typedef struct {  
    UBYTE AddrInfo[4];  
    unsigned short ExpandedNode;  
    unsigned short ObjectType;  
    unsigned short ObjectInstanceID; ;  
} XL_IPCallServerReset;
```

C++ Class

```
class XLC_IPCallServerReset: public XLC_OutboundMessage {  
public:  
    const UBYTE *getAddrInfo() const;  
    UBYTE *getAddrInfo();  
    void setAddrInfo(UBYTE *x);  
    unsigned short getExpandedNode() const;  
    void setExpandedNode(unsigned short x);  
    unsigned short getObjectType() const;  
    void setObjectType(unsigned short x)  
    unsigned short getObjectInstanceID const;  
    void setObjectInstanceID(unsigned short x)  
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x000B)
3, 4	Message Type (0x0107)	3, 4	Message Type (0x0107)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Sequence Number (same as message)
7	Extended Logical Node ID (0xFE)	7	Logical Node ID (0xFE)
8	<u>AIB</u>	8, 9	Status MSB, LSB 0x0001
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow	10-15	AIB (same as message)
	AE 0x48 Ring Communication Link 0x53 Object Type		
:	Checksum	16-19	AIB (same as message)
		:	Checksum

Reset Matrix 0x009D

SwitchKit Name ResetMatrix

Type EXS API or SwitchKit API message

Description **Reset Matrix 0x009D**

This message is used by the host to force a matrix to reset. If the host sends it to the active CSP Matrix Series 3 Card in a redundant system, it will result in a CSP Matrix Series 3 Card switchover. A switchover can also be initiated by sending a *Become Active* message to the standby CSP Matrix Series 3 Card.

Sent by Host

Related API Messages *Become Active* 0x00A1 (ActivateMatrix)

SwitchKit Code **C Structure**

```
typedef struct {
    } XL_ResetMatrix;
```

C++ Class

```
class XLC_ResetMatrix : public XLC_OutboundMessage {
public:
    };
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x0005)	1, 2	Length (0x0007)
3, 4	Message Type (0x009D)	3, 4	Message Type (0x009D)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Checksum	8, 9	Status MSB, LSB
		10	Checksum

Reset Standby Matrix

Type	SwitchKit API message
Description	Use the <i>SK_ResetStandbyMatrix</i> message to reset the secondary CSP Matrix Series 3 Card.
Sent by	Application
C Structure	<pre>typedef struct { } SK_ResetStandbyMatrix;</pre>
C++ Class	<pre>class SKC_ResetStandbyMatrix : public SKC_ToolkitMessage { public: };</pre>

Resource Attribute Configure 0x00E3

SwitchKit Name	ResourceAttributeConfig
Type	EXS API and SwitchKit API message
Description	<p>Resource Attribute Configure 0x00E3</p> <p>Use this message to configure attributes on a VDAC-ONE card, IP Network Interface Series 2 card, or a VoIP module.</p> <p>For VDAC-ONE cards, if a span/channel is active, this message waits until the call ends before updating the configuration.</p> <p>For IP Network Interface Series 2 cards, you can change the payload size and type during the call.</p>

Sent By Host

SwitchKit Code **Configuration**

```
ResourceAttributeConfig (
    Node = integer,
    Slot = integer,
    DataType = integer,
    TLVCount = integer,
    Data = byte array);
```

C Structure

```
typedef struct {
    UBYTE Slot;
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_ResourceAttributeConfig;
```

C++ Class

```
class XLC_ResourceAttributeConfig : public
    XLC_OutboundMessage {
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0XFE)	0	Frame (0XFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00E3)	3, 4	Message Type (0x00E3)
5	Reserved	5	Reserved
6	Sequence Number	6	Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow	10	Checksum
	AEs 0x01 Slot		
:	Data Type 0x00 - TLVs		

Resource Attribute Configure 0x00E3

:	Number of TLV blocks to follow
:	Data (TLVs) 0x0009 Address Element TLV (Required if TLV 0x01E2 below is included in message.) 0x0100 RTP Payload Type (VDAC-ONE) 0x0100 RTP Payload Type (IPN Series 3) 0x0100 RTP Payload Type (IPN Series 2) 0x0101 RTP Payload Size (IPN Series 2) 0x0102 RTP Silence Suppression 0x0103 RTP Echo Cancellation 0x01C2 Minimum Jitter Buffer Delay 0x01C3 Maximum Jitter Buffer Delay (IPN-2 Card) 0x01C4 Adaptation Rate 0x01C5 Fax Type Enable 0x01C7 Fax/Modem Bypass Coder Type 0x01C8 Protocol Type 0x01D0 Gateway Mode 0x01D1 RTP Payload Redundancy 0x01D2 Fax Payload Redundancy 0x01D4 Type of Service 0x01D7 Default Domain Name 0x01DB Connection Mode 0x01DF UDP Source Port Validate 0x01E1 Fax Compatibility Mode 0x01E2 RFC 2833 Enable 0x01E5 VoIP Resource Profile Assign 0x01E6 Source T.38 Port 0x01E7 Destination T.38 Port 0x01E8 Source RTCP Port 0x01E9 Destination RTCP Port 0x01EB RTP Timer Timeout (VDAC-ONE) 0x01EB Initial Media Inactivity Timeout (IP Network Interface Series 2 Card) 0x01EC Media Inactivity Detection Timer 0x01F1 RFC 2833 Dynamic Payload Type 0x2794 Destination IP Address 0x2795 Destination RTP Port 0x27B0 RTP Payload Type
:	Checksum

Resource Attribute Query 0x00E4

SwitchKit Name	ResourceAttributeQuery
Type	EXS API and SwitchKit API message
Description	<p>Resource Attribute Query 0x00E4</p> <p>This message queries either the default attributes of a module, or the current attributes of a channel involved in an IP call.</p>
Sent By	Host

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE Slot;
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_ResourceAttributeQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE Slot;
    UBYTE TLVCount;
    UBYTE Data[220];
} XL_ResourceAttributeQueryAck;
```

C++ Class

```
class XLC_ResourceAttributeQuery : public
    XLC_OutboundMessage {
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

C++ Class Response

```
class XLC_ResourceAttributeQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
```

```

UBYTE getTLVCount() const;
void setTLVCount(UBYTE x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};

```

EXS API Hex Format

Message (White)		Response (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0Xfe)	0	Frame (0XFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00E4)	3, 4	Message Type (0x00E4)
5	Reserved	5	Reserved
6	Sequence Number	6	Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow	10	AIB Same as message
	AEs 0x01 Slot	:	Number of TLV blocks
:	TLV data type (0x00)	:	TLVs 0x0100 RTP Payload Type (VDAC-ONE) 0x0100 RTP Payload Type (IPN Series 2) 0x0100 RTP Payload Type (IPN Series 3) 0x0101 RTP Payload Size (IPN Series 2)
:	Number of TLVs to follow		
:	0x0009 Address Element TLV See format choices below table	:	Checksum
:	Checksum		

Tag/Length/Value Blocks

Only one format of the Address Element Block TLV is required.

If the TLV contains the data type of Expanded Span/Channel, then the message is querying attributes of an established call.

If the TLV contains the data type of IP Address, then the message is querying default attributes of all channels associated with a particular DSP.

Message TLVs Address Element TLV Block
(for the Expanded Span/Channel AIB)

Byte	Description
0, 1	Tag: Address Element Block (0x0009)
2, 3	Length: 0x0005
4	Value: Data[0] AIB Type: Expanded Span/Channel (0x0D)
5	Value: Data[1] AIB Length (0x03)
6	Value: Data[2] Logical Span (MSB)
7	Value: Data[3] Logical Span (LSB)
8	Value: Data[4] Channel

Address Element TLV Block
(for the IP Address AIB)

Byte	Description
0, 1	Tag: Address Element Block (0x0009)
2, 3	Length: 0x0006
4	Value: Data[0] AIB Type: (0x3E: IP Address)
5	Value: Data[1] AIB Length (0x04)
6	Value: Data[2] IP (MSB)
7	Value: Data[3] IP
8	Value: Data[4] IP
9	Value: Data[5] IP (LSB)

Resource Connect 0x0127

SwitchKit Name ResourceConnect

Type EXS and SwitchKit API message

Description **Resource Connect 0x0127**

Use this message to connect to a resource.

Connect to a Child Conference

Use the Channel AIB and the Child Conference AIB. The Channel AIB indicates the Channel to be moved from the parent conference to the child conference. The Child Conference ID AIB indicates both the parent and child conference IDs.

NOTE: For the SwitchKit code, you must specify the span and channel before you specify the conference ID.

Sent by Host application

Related API Messages *Resource Create* (0x0124)
Resource Modify (0x125)
Resource Delete (0x126)

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_ResourceConnect;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[219];
} XL_ResourceConnectAck;
```

C++ Class

```
class XLC_ResourceConnect: public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
```

```

void setAddrInfo(UBYTE *x);
UBYTE getChannelSlot() const;
void setChannelSlot(UBYTE x)
XBYTE getConferenceID() const;
void setConferenceID(XBYTE x);
XBYTE getParentConferenceID() const;
void setParentConferenceID(XBYTE x);
XBYTE getChildConferenceID() const;
void setChildConferenceID(XBYTE x);
XBYTE getSpan() const;
void setSpan(XBYTE x);
UBYTE getChannel() const;
void setChannel(UBYTE x);
UBYTE getDataType() const;
void setData(UBYTE x);
UBYTE getTLVCount() const ;
void setTLVCount(UBYTE x) ;
const UBYTE *getData() const ;
UBYTE *getData();
void setData(UBYTE *x) ;
};

```

C++ Class Response

```
class XLC_ResourceConnectAck : public
```

```
XLC_AcknowledgeMessage {
```

```
public:
```

```

    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getDataType() const;
    void setData(UBYTE x);
    UBYTE getTLVCount() const ;
    void setTLVCount(UBYTE x) ;
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x0127)	3, 4	Message Type (0x0127)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status (MSB, LSB)
8 : : :	AIB	10 . . .	AIB (same as message)
	Address Method 0x00 - Individual AEs		
	Number of AEs to follow		
AEs When using this message for conferencing: Resource A 0x0D Channel Resource B 0x55 Conference ID 0x45 Child Conference ID When using this message for faxing or Echo Cancel: 0x0D Channel 0x01 Slot		:	Data Type 0x00 TLVs
:	Data Type 0x00 TLVs	:	Number of TLVs to Follow
:	Data Type 0x00 TLVs	:	Mandatory Return TLVs 0x0602 Resource Type
:	Number of TLVs to Follow	:	Checksum

Mandatory TLVs

0x0602 Resource Type
(This is the B Resource Type)

If the Resource Type is Send or Receive Fax:

0x05E0 File ID

0x05E2 File Location

If the Resource Type is set to Send SMS (0x010A) or Receive SMS (0x010B):

0x0694 SMS Data

If the Resource Type is set to Receive FSK (0x010D):

0x0690 FSK Data

Optional TLVs

If AIB is Conference and Resource Type is Conference:

0x0604 DTMF Clamping/Filtering Enable (Default is Disabled)

0x0607 Output Gain Control (Default is 0db)

0x0608 Noise Gating Enable (Default is Disabled)

0x060A Echo Suppression Enable (Default is Disabled)

0x060C Automatic Gain Control Enable (Default is Disabled)

0x0612 Connection Type

0x060F Conference Failure Behavior

0x068B Input Gain Control

0x068D Transit Connection Mode

Optional TLVs

If AIB is Child Conference ID and Resource Type is Conference:

0x0604 DTMF Clamping/Filtering Enable (Default is Disabled)

0x0607 Output Gain Control (Default is 0db)

0x0608 Noise Gating Enable (Default is Disabled)

0x060A Echo Suppression Enable (Default is Disabled)

0x060C Automatic Gain Control Enable (Default is Disabled)

	<p><u>Optional TLVs for Echo Cancel Parameters</u></p> <p>0x0673 Echo Cancel Tap Length 0x0674 Echo Cancel NLP Type 0x0675 Echo Cancel ADAPT 0x0676 Echo Cancel Bypass 0x0677 Echo Cancel G.176 Modem Answer Detection 0x0678 Echo Cancel NLP Threshold 0x0679 Echo Cancel CNG Noise Threshold 0x067E Echo Cancel Comfort Noise Level</p> <p><u>Optional TLVs for PVD/AMD</u></p> <p>0x0619 PVD Parameters 0x061B AMD Reports</p> <p><u>Optional TLVs for Send Fax or Receive Fax</u></p> <p>0x0641 Header Parameter Format 0x0642 T.30 Control Parameter Max Value 0x0643 T.30 Control Parameter Transmit Level 0x0644 T.30 Control Parameter ECM Enabled 0x0645 T.30 Control Parameter Local Session ID 0x0648 Receive Resolution Type 0x064C Receive Page Size 0x0651 Receive Enable ECM 0x0652 Receive Add Header 0x0655 Receive Timeout 0x0657 Receive Terminal ID 0x0661 Transmit Enable ECM 0x0662 Transmit Add Header 0x0664 Transmit Enable CNG 0x0668 Transmit dbm Level 0x066A Transmit Terminal ID 0x068C FAX Page Range</p> <p><u>Optional TLV for Send SMS (0x010A) or Receive SMS (0x010B)</u></p> <p>0x0695 FSK Modem Type</p>
:	Checksum

Resource Create 0x0124

SwitchKit Name Resource Create

Type EXS and SwitchKit API message

Description **Resource Create 0x0124**

Use this message to create a processing resource in the CSP.

Conference

To create a Unified conference, you send the following:

AIB: DSP Chip

Resource Type TLV: Conference

TLVs

Channel/DSP Pool

Conference Size

Monitor Conference Enable (to create a Monitor Conference)

Any optional conference TLVs

The Conference ID is returned in the response.

Child Conference

To create a Child Conference, you send the following:

AIB: Child Conference ID

TLVs

Resource Type: Child Conference

Any optional conference TLVs

The Child Conference ID is returned in the response.

Sent by Host application

Related API Messages *Resource Modify (0x125)*

Resource Delete (0x126)

Resource Connect (0x127)

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_ResourceCreate;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[219];
} XL_ResourceCreateAck;
```

C++ Class

```
class XLC_ResourceCreate : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    UBYTE getDSPChip() const;
    void DSPChip(UBYTE x);
    UBYTE getDSPSlot() const;
    void setDSPSlot(UBYTE x);
    UBYTE getDSPSIMM() const;
    void setDSPSIMM(UBYTE x);
    UBYTE getDataType() const;
    void setDataType(UBYTE x) ;
    UBYTE getTLVCount() const ;
    void setTLVCount(UBYTE x) ;
    const UBYTE *getData() const ;
    UBYTE *getData() ;
    void setData(UBYTE *x) ;
}
```

C++ Class Response

```
class XLC_ResourceCreateAck: public
XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    UBYTE getDSPChip() const;
    void DSPChip(UBYTE x);
    UBYTE getDSPSlot() const;
    void setDSPSlot(UBYTE x);
    UBYTE getDSPSIMM() const;
```

```
void setDSPSIMM(UBYTE x);  
UBYTE getDataType() const;  
void setDataType(UBYTE x) ;  
UBYTE getTLVCount() const ;  
void setTLVCount(UBYTE x) ;  
const UBYTE getDataType() const;  
UBYTE *getData() ;  
void setDataType(UBYTE x);  
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x0124)	3, 4	Message Type (0x0124)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status (MSB, LSB) 0x0001 Invalid TLV Data Software can't find the TLV Data Buffer. This can also occur if the Data for a TLV is out of range. 0x0003 Invalid number of TLVs There are no TLVs in the message. 0x0004 Invalid TLV Length The TLV length is different from what is expected. 0x0006 Invalid TLV Unknown TLV 0x000D Mandatory TLVs missing One or more mandatory TLVs are missing
8	<u>AIB</u>	10	AIB (same as message)
:	Address Method	:	
	0x00 - Individual AEs		
	Number of AEs to follow		
	AE		
	0x22 DSP Chip		
	0x45 Child Conference ID		
:	Data Type	:	Data Type
	0x00 TLVs		0x00 TLVs

Resource Create 0x0124

10	Number of TLVs to Follow	:	Number of TLVs to Follow
:	<p>TLVs</p> <p>Mandatory for all: 0x0602 Resource Type</p> <p>Mandatory if Resource Type is Conference: 0x0603 Channel/DSP Pool 0x0613 Conference Size</p> <p>Optional if Resource Type is Conference or Child Conference: (Note: if Resource Type is Child Conference, only TLVs 0x0607 - 0x060E can be used) (Note: for these TLVs to work, the corresponding function must already be enabled on the card, using the DSP SIMM Configure message):</p> <p>0x05E5 Barge In (Enable) 0x0604 DTMF Clamping/Filtering Enable (default is Disabled) 0x0605 EXS Conferencing Encoding Type (default Mixed) 0x0606 Monitor Conference Enable (default is Disabled) 0x0607 Output Gain Control (default 0db) 0x0608 Noise Gating Enable (default Disabled) 0x0609 Noise Gate Parameters 0x060A Echo Suppression Enable (default Disabled) 0x060B Echo Suppression Parameters 0x060C Automatic Gain Control Enable (default Disabled) 0x060D Automatic Gain Control Input Level 0x060E Automatic Gain Control Parameters 0x060F Conference Failure Behavior (default Purge) 0x068B Input Gain Control 0x068D Transit Connection Mode</p> <p>For Conferencing: The Conference can have listen-only channels connected to it.</p> <p>Optional if Resource Type is File Playback 0x05E5 Barge In (Enable) The File will play continuously until no timeslots are listening to it. Multiple timeslots can listen to the same File payout</p>	:	<p>TLVs</p> <p>Mandatory: 0x0602 Resource Type</p> <p>Mandatory if Resource Type is Conference or Child Conference: 0x0610 Conference ID</p>
:	Checksum	:	Checksum

Resource Delete 0x0126

SwitchKit Name	ResourceDelete
Type	EXS API and SwitchKit API message
Description	<p>Resource Delete 0x0126</p> <p>Use this message to delete a processing resource.</p>

Deleting a Conference

If forced deletion of a child conference is specified, the child conference is deleted immediately and the parties are connected back to the parent conference. If graceful deletion of a child conference is specified, then the child conference is deleted only after all the parties connected to it are disconnected. If the Forced Delete TLV is not used, graceful deletion is the default behavior.

When you use this message to delete a child conference, you must use the Child Conference ID AIB to address the child conference that needs to be deleted. If the child conference ID is specified as FFFF, then all the child conferences for the particular parent conference are deleted.

Sent by Host application

Related API Messages

- Resource Create (0x0124)*
- Resource Modify (0x125)*
- Resource Connect (0x127)*
- Resource Delete Indication (0x0129)*

SwitchKit Code C Structure

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_ResourceDelete;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[219];
} XL_ResourceDeleteAck;
```

C++ Class

```

class XLC_ResourceDelete : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);

    // Extended addressing functions
    // Conference ID AIB functions
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    void setChildConferenceID(XBYTE x);
    UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const ;
    void setTLVCount(UBYTE x) ;
    const UBYTE *getData() const ;
    UBYTE *getData();
    void setData(UBYTE *x) ;

```

C++ Class Response

```

class XLC_ResourceDeleteAck : public
XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const ;
    void setTLVCount(UBYTE x) ;
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x0126)	3, 4	Message Type (0x0126)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8, 9	Status (MSB, LSB)
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow	10	AIB (same as message)
	AE	:	Data Type
	0x55 Conference ID		0x00 TLVs
	0x45 Child Conference ID		
:	Data Type	:	Number of TLVs to Follow
	0x00 TLVs		
:	Number of TLVs to Follow	:	TLVs
			Mandatory Return TLVs
			0x0602 Resource Type
:	TLVs	:	Checksum
	Mandatory TLV: 0x0602 Resource Type		
	Optional TLV if AE is Conference and Resource Type is Conference: 0x0611 Forced Flag (Default is Graceful)		
	Optional TLV if AE is Child Conference and Resource Type is Child Conference: 0x0611 Forced Flag (Default is Graceful)		
:	Checksum		

Resource Delete Indication 0x0129

SwitchKit Name

Type EXS API and SwitchKit API message

Description **Resource Delete Indication 0x0129**

This message indicates that a resource has been deleted.

Conferencing

When a child conference has been deleted, this message uses the Child Conference ID AIB as well as the mandatory AIB Container TLV (0xE003).

Sent by CSP

Related API Messages

- Resource Create* (0x0124)
- Resource Modify* (0x125)
- Resource Delete* (0x0126)
- Resource Connect* (0x127)

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_ResourceDeleteIndication;
```

C++ Class

```
class XLC_ResourceDeleteIndication : public
    XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);

    // Extended addressing functions
    // Conference ID AIB functions
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    void setChildConferenceID(XBYTE x);
    UBYTE getDataType() const;
    void setDataType(UBYTE x);
```

```

UBYTE getTLVCount() const ;
void setTLVCount(UBYTE x) ;
const UBYTE *getData() const ;
UBYTE *getData();
void setData(UBYTE *x) ;
    
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x0129)	3, 4	Message Type (0x0129)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Address Information Block 0x45 Child Conference ID	8	Checksum
:	Data Type 0x00 (2-byte global TLVs) <u>Mandatory TLVs</u> 0xE003 AIB Container TLV - if indicating deletion of a child conference		
:	Number of TLVs to Follow		
:	:		
:	Checksum		

Resource Disconnect 0x0128

SwitchKit Name ResourceDisconnect

Type EXS and SwitchKit API message

Description **Resource Disconnect 0x0128**

Use this message to disconnect a processing resource, which was previously connected using the *Resource Connect* message.

FAX

Use this message to disconnect a fax channel in order to interrupt fax processing on that channel.

Conferencing

Use this message to move a channel from a Child Conference back into the Parent Conference. Use the Channel AIB and the Child Conference AIB. The Channel AIB indicates the Channel to be moved from the child conference to the parent conference. The Child Conference ID AIB both the parent and child conference IDs.

Sent by Host application

Related API Messages

- Resource Create* (0x0124)
- Resource Modify* (0x0125)
- Resource Delete* (0x0126)
- Resource Connect* (0x0127)
- Resource Query* (0x012C)

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_ResourceDisconnect;
};
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[219];
} XL_ResourceDisconnectAck;
```

};

C++ Class

```

class XLC_ResourceDisconnect : public XLC_OutboundMessage
{
public:
    const UBYTE *getAddrInfo() const ;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    XBYTE getSpan() const ;
    void setSpan(XBYTE x) ;
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getDataType() const ;
    void setData(UBYTE x);
    UBYTE getTLVCount() const ;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData() ;
    void setData(UBYTE *x)};
};

```

C++ Class Response

```

class XLC_ResourceDisconnectAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getDataType() const
    void setData(UBYTE x);
    UBYTE getTLVCount() const ;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() ;
    UBYTE *getData();
    void setData(UBYTE *x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x0128)	3, 4	Message Type (0x0128)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status (MSB, LSB)
8	<u>AIB</u>	10	AIB (same as message)
:	Address Method		
	0x00 - Individual AEs		
	Number of AEs to follow		
	AE		
	0x0D Channel		
	0x45 Child Conference ID		
:	Data Type	:	Data Type
	0x00 TLVs		0x00 TLVs
:	Number of TLVs to Follow	:	Number of TLVs to Follow
:	<u>Mandatory TLVs</u>	:	:
	0x0602 Resource Type		
	(This is the B Resource Type)		
	If the Resource Type is Send or Receive Fax:		
	0x05E0 File ID		
	0x05E0 File ID 0x05E2 File Location		
	<u>Optional TLVs</u>		
	If AIB is Child Conference ID and Resource Type is		
	Conference:		
	0x0604 DTMF Clamping/Filtering Enable (Default is Disabled)		
	0x0607 Output Gain Control (Default is 0db)		
	0x0608 Noise Gating Enable (Default is Disabled)		
	0x060A Echo Suppression Enable (Default is Disabled)		
	0x060C Automatic Gain Control Enable (Default is Disabled)		
:	Checksum	:	Checksum

Resource Information Notify 0x0141

SwitchKit Name	ResourceInfoNotify
Type	EXS and SwitchKit API message
Description	Resource Information Notify 0x0141 Use this message to connect to a resource.

Connecting to a different resource types

Use this generic message to specify different Resource Types. When used for Frequency Shift Keying (FSK), the mandatory Resource Type TLV 0x0602 will be set to either Send SMS (0x010C) or Receive SMS (0x010D) and from the Switch to notify the Host of the SMS data received by the DSP Series 2 card.

Sent by Switch application

Related API Messages

- Resource Create* (0x0124)
- Resource Modify* (0x125)
- Resource Delete* (0x126)

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_ResourceInfoNotify;
```

C++ Class

```
class XLC_ResourceInfoNotify: public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const ;
    void setTLVCount(UBYTE x) ;
    const UBYTE *getData() const ;
    UBYTE *getData();
    void setData(UBYTE *x) ;
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)			
Byte	Field Description	Byte	Field Description		
0	Frame (0xFE)	0	Frame (0xFE)		
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)		
3, 4	Message Type (0x0127)	3, 4	Message Type (0x0127)		
5	Reserved (0x00)	5	Reserved (0x00)		
6	Sequence Number	6	Same Sequence Number		
7	Logical Node ID	7	Logical Node ID		
		8, 9	Status (MSB, LSB)		
8 : : :	<u>AIB</u> Address Method 0x00 - Individual AEs	10 . . .	AIB (same as message)		
	Number of AEs to follow			:	Data Type 0x00 TLVs
	AEs When using this message for conferencing: Resource A 0x0D Channel			:	Number of TLVs to Follow
:	Data Type 0x00 TLVs	:	Mandatory Return TLVs 0x0602 Resource Type		
:	Number of TLVs to Follow	:	Checksum		
:	<u>Mandatory TLVs</u> 0x0602 Resource Type (This is the A Resource Type) If the Resource Type is set to Send SMS (0x010A) or Receive SMS (0x010B) 0x0694 SMS Data If the Resource Type is set to Receive FSK (0x010D): 0x0690 FSK Data				
	<u>Optional TLVs</u>				
:	Checksum				

Resource Modify 0x0125

SwitchKit Name	Resource Modify
Type	EXS and SwitchKit API message
Description	Resource Modify 0x0125 Use this message to change parameters for a processing resource in the CSP.
Sent by	Host application
Related API Message	<i>Resource Create</i> (0x0124) <i>Resource Delete</i> (0x126) <i>Resource Connect</i> (0x127)

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_ResourceModify;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[219];
} XL_ResourceModifyAck;
```

C++ Class

```
class XLC_ResourceModify : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
};
```

```
    UBYTE *getData() const ;  
    void setData(UBYTE *x) ;  
};
```

C++ Class Response

```
class XLC_ResourceModifyAck : public  
XLC_AcknowledgeMessage {  
public:  
    unsigned short getStatus() const;  
    void setStatus(unsigned short x);  
    const UBYTE *getAddrInfo() const;  
    UBYTE *getAddrInfo();  
    void setAddrInfo(UBYTE *x);  
    XBYTE getConferenceID() const;  
    void setConferenceID(XBYTE x);  
    XBYTE getSpan() const;  
    void setSpan(XBYTE x);  
    UBYTE getChannel() const;  
    void setChannel(UBYTE x);  
    UBYTE getDataType() const;  
    void setDataType(UBYTE x);  
    UBYTE getTLVCount() const ;  
    void setTLVCount(UBYTE x) ;  
    const UBYTE *getData() const;  
    UBYTE *getData();  
    void setData(UBYTE *x);  
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x0125)	3, 4	Message Type (0x0125)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB)
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow	10	AIB (same as message)
	AEs 0x0D Channel 0x55 Conference ID 0x45 Child Conference ID	:	Data Type 0x00 TLVs
:	Data Type 0x00 TLVs	:	Number of TLVs to Follow
:	Number of TLVs to Follow	:	Mandatory Return TLVs 0x0602 Resource Type

Resource Modify 0x0125

:	<p>TLVs</p> <p>Mandatory TLVs</p> <p>0x0602 Resource Type</p> <p>If the Resource Type is set to Send SMS (0x010A) or Receive SMS (0x010B):</p> <p>0x0694 SMS Data</p> <p>Optional TLVs</p> <p>Optional TLVs if AIB is Conference and Resource Type is Conference or Child Conference:</p> <p>0x0604 DTMF Clamping/Filtering Enable (Default is Disabled)</p> <p>0x0607 Output Gain Control (Default is 0db)</p> <p>0x0608 Noise Gating Enable (Default is disabled)</p> <p>0x060A Echo Suppression Enable (Default is disabled)</p> <p>0x060C Automatic Gain Control Enable (Default is disabled)</p> <p>Optional TLVs if AIB is Channel and Resource Type is Conference or Child Conference:</p> <p>0x0604 DTMF Clamping/Filtering Enable</p> <p>0x0607 Output Gain Control</p> <p>0x0608 Noise Gating Enable</p> <p>0x060A Echo Suppression Enable</p> <p>0x060C Automatic Gain Control Enable</p> <p>0x068B Input Gain Control</p> <p><u>Optional TLVs for Echo Cancel Parameters</u></p> <p>0x0673 Echo Cancel Tap Length</p> <p>0x0674 Echo Cancel NLP Type</p> <p>0x0675 Echo Cancel ADAPT</p> <p>0x0676 Echo Cancel Bypass</p> <p>0x0677 Echo Cancel G.176 Modem Answer Detection</p> <p>0x0678 Echo Cancel NLP Threshold</p> <p>0x0679 Echo Cancel CNG Noise Threshold</p> <p>0x067A Echo Cancel H Register Reset</p> <p><u>Optional TLV for Send SMS (0x010A) or Receive SMS (0x010B)</u></p> <p>0x0695 FSK Modem Type</p>	:	Checksum
:			Checksum

Resource Query 0x012C

SwitchKit Name ResourceQuery

Type EXS and SwitchKit API message

Description **Resource Query 0x012C**

Use this message to query different configuration parameters that have been set for a particular channel (or any other entity used as an A-resource) using the *Resource Connect* (0x0127), *Resource Create* (0x0124), or *Resource Modify* (0x0125) messages.

You can use this message to query fax configuration parameters set for a particular channel on a per-call basis. These optional fax configuration parameters would have been configured while initiating a fax call, using the *Resource Connect* message (0x0127).

Note: To query configuration parameters that have been set at the card level using the *Generic Card Configure* (0x0122) message, use the *Generic Card Query* message (0x023).

Sent by Host application

Related API Messages *Resource Create* (0x0124)
Resource Modify (0x0125)
Resource Delete (0x0126)
Resource Connect (0x0127)
Resource Query (0x012C)

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_ResourceQuery;
};
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[219];
} XL_ResourceQueryAck;
};
```

C++ Class

```

class XLC_ResourceQuery : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const ;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getSpan() const ;
    void setSpan(XBYTE x) ;
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getDataType() const ;
    void setData(UBYTE x);
    UBYTE getTLVCount() const ;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData() ;
    void setData(UBYTE *x)};
};

```

C++ Class Response

```

class XLC_ResourceQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getDataType() const
    void setData(UBYTE x);
    UBYTE getTLVCount() const ;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() ;
    UBYTE *getData();
    void setData(UBYTE *x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x0128)	3, 4	Message Type (0x0128)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status (MSB, LSB)
8 : :	<u>AIB</u> Address Method	10	AIB (same as message)
	Number of AEs to follow	:	Data Type 0x00 TLVs
	If Resource Type is Send or Receive Fax: 0x0D Channel	:	Number of TLVs to follow

Resource Query 0x012C

:	<p>Data Type 0x00 TLVs</p>	:	<p>TLVs</p> <p>Mandatory TLVs</p> <p>0x0602 Resource Type</p> <p>If TLV 0x066D is sent with a Query Type of 0x0001 (T.30 control parameters), the following TLVs are returned:</p> <ul style="list-style-type: none"> 0x0641 Header Parameter Format 0x0642 T.30 Control Parameter Max Value 0x0643 T.30 Control Parameter Transmit Level 0x0644 T.30 Control Parameter ECM Enabled 0x0645 T.30 Control Parameter Local Session ID <p>If TLV 0x066D is sent with a Query Type of 0x0002 (Receive Parameters), the following TLVs are returned:</p> <ul style="list-style-type: none"> 0x0648 Receive Resolution Type 0x0648 Receive Resolution Type 0x0649 Receive Encoding Type 0x064B Receive Bad Line 0x064C Receive Page Size 0x0651 Receive Enable ECM 0x0652 Receive Add Header 0x0654 Receive Line Error Threshold 0x0655 Receive Timeout 0x0657 Receive Terminal ID <p>If TLV 0x066D is sent with a Query Type of 0x0004 (Transmit parameters), the following TLVs are returned:</p> <ul style="list-style-type: none"> 0x065A Transmit Modem Type 0x065B Transmit Resolution Type 0x065C Transmit Page Size 0x0661 Transmit Enable ECM 0x0662 Transmit Add Header 0x0664 Transmit Enable CNG 0x0666 Transmit Timeout 0x0668 Transmit dbm Level 0x066A Transmit Terminal ID
---	--------------------------------	---	---

Resource Query 0x012C

:	Number of TLVs to Follow	:	Checksum
	<p><u>TLVs</u></p> <p><u>Mandatory TLVs</u></p> <p>0x0602 Resource Type</p> <p>If Resource type is Send Fax or Receive Fax: 0x066D Fax Configuration Query Type</p> <p>OR one of the following Fax Config TLVs:</p> <p>0x0641 Header Parameter Format 0x0642 T.30 Control Parameter Max Value 0x0643 T.30 Control Parameter Transmit Level 0x0644 T.30 Control Parameter ECM Enabled 0x0645 T.30 Control Parameter Local Session ID 0x0648 Receive Resolution Type 0x064C Receive Page Size 0x0651 Receive Enable ECM 0x0652 Receive Add Header 0x0655 Receive Timeout 0x0657 Receive Terminal ID 0x0661 Transmit Enable ECM 0x0662 Transmit Add Header 0x0664 Transmit Enable CNG 0x0668 Transmit dbm Level 0x066A Transmit Terminal ID</p>		
:	Checksum		

Resource Release Indication 0x012E

SwitchKit Name ResourceReleaseIndication

Type EXS and SwitchKit API message

Description **Resource Release Indication 0x012E**

Use this message to connect to a resource.

Connecting to a different resource types

Use this generic message to specify different Resource Types. When used for Frequency Shift Keying (FSK), the mandatory Resource Type TLV 0x0602 will be set to either Send SMS (0x010C) or Receive SMS (0x010D) and sent from the Switch to the Host to indicate a SMS call release at the Data Link Layer.

Sent by Switch application

Related API Messages *Resource Create* (0x0124)
Resource Modify (0x125)
Resource Delete (0x126)

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_ResourceReleaseIndication;
```

C++ Class

```
class XLC_ResourceReleaseIndication: public
    XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const ;
    void setTLVCount(UBYTE x) ;
    const UBYTE *getData() const ;
    UBYTE *getData();
    void setData(UBYTE *x) ;
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x012E)	3, 4	Message Type (0x012E)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status (MSB, LSB)
8 : : :	<u>AIB</u> Address Method 0x00 - Individual AEs	10 . . .	AIB (same as message)
	Number of AEs to follow		: Data Type 0x00 TLVs
	AEs When using this message for conferencing: Resource A 0x0D Channel		: Number of TLVs to Follow
:	Data Type 0x00 TLVs	:	Mandatory Return TLVs 0x0602 Resource Type
:	Number of TLVs to Follow	:	Checksum
:	<u>Mandatory TLV</u> 0x0602 Resource Type (This is the A Resource Type)		
	If the Resource Type is set to Send SMS (0x010A) or Receive SMS (0x010B) 0x0694 SMS Data		
:	Checksum		

Resource Release Request 0x012D

SwitchKit Name ResourceReleaseRequest

Type EXS and SwitchKit API message

Description **Resource Release Request 0x012D**
Use this message to connect to a resource.

Connecting to a different resource types

Use this generic message to specify different Resource Types. When used for Frequency Shift Keying (FSK), the mandatory Resource Type TLV 0x0602 will be set to either Send SMS (0x010C) or Receive SMS (0x010D) and sent from the Host to the switch to release the Data Link Layer connection in a SMS call.

Sent by Host application

Related API Messages *Resource Create* (0x0124)
Resource Modify (0x125)
Resource Delete (0x126)

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_ResourceReleaseRequest;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE AddrInfo[30];
    UBYTE DataType;
    UBYTE TLVCount;
    UBYTE Data[219];
} XL_ResourceReleaseRequestAck;
```

C++ Class

```
class XLC_ResourceReleaseRequest: public
    XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
```

```
XBYTE getParentConferenceID() const;
void setParentConferenceID(XBYTE x);
XBYTE getChildConferenceID() const;
void setChildConferenceID(XBYTE x);
XBYTE getSpan() const;
void setSpan(XBYTE x);
UBYTE getChannel() const;
void setChannel(UBYTE x);
UBYTE getDataType() const;
void setDataType(UBYTE x);
UBYTE getTLVCount() const ;
void setTLVCount(UBYTE x) ;
const UBYTE *getData() const ;
UBYTE *getData();
void setData(UBYTE *x) ;
};
```

C++ Class Response

```
class XLC_ResourceReleaseRequestAck : public
XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getDataType() const;
    void setDataType(UBYTE x);
    UBYTE getTLVCount() const ;
    void setTLVCount(UBYTE x) ;
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x0127)	3, 4	Message Type (0x0127)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number

Resource Release Request 0x012D

7	Logical Node ID	7	Logical Node ID
		8, 9	Status (MSB, LSB)
8	<u>AIB</u> Address Method 0x00 - Individual AEs	10 . . .	AIB (same as message)
	Number of AEs to follow	:	Data Type 0x00 TLVs
	AEs When using this message for conferencing: Resource A 0x0D Channel	:	Number of TLVs to Follow
:	Data Type 0x00 TLVs	:	Mandatory Return TLVs 0x0602 Resource Type
:	Number of TLVs to Follow	:	Checksum
	<u>Mandatory TLV</u> 0x0602 Resource Type (This is the A Resource Type) If the Resource Type is set to Send SMS (0x010A) or Receive SMS (0x010B) 0x0694 Short Message Service (SMS) Data <u>Optional TLV for Send SMS (0x010A) or Receive SMS (0x010B)</u> 0x0695 FSK Modem Type : Checksum		

Ring Status Query 0x0071

SwitchKit Name	RingStatusQuery
Type	EXS API and SwitchKit API message
Description	Ring Status Query 0x0071 Use this message to query either the ring status or the ring configuration.
To Query Ring Configuration	Use the Ring Configuration Query address type in the AIB to query the ring configuration. See “Response Data” for the format of the data returned by the CSP.
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE AddrInfo[253];
} XL_RingStatusQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE Slot;
    UBYTE Data[221];
} XL_RingStatusQueryAck;
```

C++ Class

```
class XLC_RingStatusQuery : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
};
```

C++ Class Response

```
class XLC_RingStatusQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x0071)	3, 4	Message Type (0x0071)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8, 9	Status MSB, LSB
:	Address Method		
	Number of AEs to follow	10	AIB (same as message)
	AEs	:	:
:	Checksum	:	Data
		:	:
		:	Checksum

**EXS API Hex Format -
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x0071)	3, 4	Message Type (0x0071)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow	10	AIB (same as message)
	AE	:	
	0x01 Slot		
	0x3D Ring Configuration Query		
:	Checksum		

:	<p>Data (also see Response Data section below this table) For a <i>Ring Configuration Query</i>, see the information below this table For a <i>Ring Status Query</i>, see the information directly below:</p> <p>Data[0] Logical Ring ID Data[1] Ring Status This field indicates the status of the ring to which this node belongs. 0x00 In Service: Ring capable of passing data 0x01 Ring Initializing: Validating integrity of the ring 0x02 Connected: Adjacent nodes are physically connected and seeing light 0xFE RIC Failure State: Must cycle the ring service state or run diagnostics 0xFF Out of Service: Ring is down and unable to pass data</p> <p>Data[2] More Status This field indicates the ring status information whenever the ring unexpectedly goes down. 0x00 EXNET Card Reset: Host has taken ring out-of-service (normal reset) 0x01 Ring Configuration Incomplete 0x02 Waiting for Addition 0x05 Unsupported Ring Mod 0xF6 EXNET Card In Failed State: EXNET Card has reset too many times 0xF7 EXNET Card Reset: EXNET Card configuration information changed 0xF8 EXNET Card Reset: Remote node timed out 0xF9 EXNET Card Timed Out During Ring Initialization 0xFA Internal Diagnostics Failed 0xFB EXNET Card Isolated From Ring 0xFC Ring Out of Service: Internal failure 0xFE Not Receiving Light 0xFF Ring Out of Service</p> <p>Data[3] EXNET Port A Status This field indicates the status of EXNET port A. 0x00 Normal 0x01 Loop Back</p> <p>Data[4] EXNET Port B Status This field indicates the status of EXNET port B. 0x00 Normal 0x01 Loop Back</p> <p>Data[5] Master Status This field indicates whether or not this node is the master. 0x00 Not Master 0x01 Master</p> <p>Data[6] Transmit Mode This field indicates status of the transmit and receive mode. 0x00 Receive Only 0x01 Transmit/Receive</p>
---	---

	<p>Data[7] Source Packet Number This field indicates the ID number of the source packet. 0x00–0x1F Source Packet Number 0xFF Source Packet Number Unassigned</p> <p>Data[8] Master Arbitration Flag This field indicates whether or not a node is allowed to become EXNET Ring Master: 0x01 This node is allowed to participate in EXNET Ring Master Arbitration. 0x00 This node is not allowed to participate in EXNET Ring Master Arbitration.</p> <p>Data[9] Redundancy State This field indicates redundancy state of the RICs: 0x00 No Redundancy 0x01 Active RIC 0x02 Standby RIC 0xFF Role Not Yet Defined</p> <p>Data[10] Current Ring Mode This field indicates the Ring Mode: 0x00 Normal Ring Mode 0x01 Enhanced Fault Tolerance Ring Mode</p>
:	Checksum

Response Data Ring Configuration Query

When querying the ring configuration, the following Data is returned.

Response Fields For Ring Configuration Query
Data[0] Number of Rings Configured
Data[1] Number of Unconfigured Nodes
Data[2] Logical Ring ID
Data[3] Ring Service State
Data[4] Number of Nodes on ring
Data[5] Ring Master Node ID
Data[6-9] Physical Node ID
Data[10-13] IP Address
Data[14] Logical Node ID
Data[15] Load Major Rev.
Data[16] Load Minor Rev.
Data[17] Load Build Number
Data[18] Transmit/Receive Mode

- Data[1] through Data[18] are repeated for each ring.
- Data[6] through Data[18] are repeated for each node on the ring.

Ring Status Report 0x0072

SwitchKit Name	RingStatusReport
Type	EXS API and SwitchKit API message
Description	<p>Ring Status Report 0x0072</p> <p>This message informs the host of a ring's status after its EXNET® status changes.</p>
Sent by	CSP
SwitchKit Code	C Structure

```
typedef struct {
    UBYTE Slot;
    UBYTE RingID;
    UBYTE RingStatus;
    UBYTE MoreStatus;
    UBYTE PortAStatus;
    UBYTE PortBStatus;
    UBYTE MasterStatus;
    UBYTE TransmitMode;
    UBYTE SourcePacketNum;
    UBYTE MasterConfigurable;
    UBYTE RedundancyState;
    UBYTE RingMode;
} XL_RingStatusReport;
```

C++ Class

```
class XLC_RingStatusReport : public XLC_InboundMessage {
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getRingID() const;
    void setRingID(UBYTE x);
    UBYTE getRingStatus() const;
    void setRingStatus(UBYTE x);
    UBYTE getMoreStatus() const;
    void setMoreStatus(UBYTE x);
    UBYTE getPortAStatus() const;
    void setPortAStatus(UBYTE x);
    UBYTE getPortBStatus() const;
    void setPortBStatus(UBYTE x);
    UBYTE getMasterStatus() const;
    void setMasterStatus(UBYTE x);
    UBYTE getTransmitMode() const;
    void setTransmitMode(UBYTE x);
    UBYTE getSourcePacketNum() const;
    void setSourcePacketNum(UBYTE x);
    UBYTE getMasterConfigurable() const;
    void setMasterConfigurable(UBYTE x);
    UBYTE getRedundancyState() const;
```

```
void setRedundancyState(UBYTE x);
UBYTE getRingMode() const;
void setRingMode(UBYTE x);
};
```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0005)
3, 4	Message Type (0x0072)	3, 4	Message Type, MSB (0x0072)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8	Checksum
:	Address Method		
	Number of AEs to follow		
	AEs		
:	Logical Ring ID		
:	Ring Status		
:	More Status		
:	EXNET® Port A Status		
:	EXNET® Port B Status		
:	Master Status		
:	Transmit Mode		
:	Source Packet Number		
:	Master Arbitration Flag		
:	Checksum		

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0005)
3, 4	Message Type (0x0072)	3, 4	Message Type (0x0072)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8	Checksum
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow		
	AE 0x01 Slot		
:	Logical Ring ID		
:	Ring Status The status of the ring to which this node belongs. 0x00 In Service: Ring capable of passing data 0x01 Ring Initializing: Validating integrity of the ring 0x02 Connected: Adjacent nodes are physically connected and seeing light 0xFE RIC Failure State: Must cycle the ring service state or run diagnostics 0xFF Out of Service: Ring is down and unable to pass data NOTE: Ring B is a backup ring, and the status indicates that the ring is initializing until the ring is brought into service due to the failure of Ring A.		
:	More Status This field indicates further status information. 0x00 None 0x01 Ring Configuration Incomplete 0x02 Waiting for Addition 0x05 Unsupported Ring Mode 0xF6 EXNET® Card has reset too many times 0xF7 Ring Mastership Configuration Changed 0xF8 Remote node timed out 0xF9 Timed out during ring initialization 0xFA Internal Diagnostics Failed 0xFB EXNET® Card isolated from ring 0xFC Internal Failure 0xFE Not Receiving Light 0xFF Ring Out of Service Unknown Failure		
:	EXNET® Port A Status These fields indicates the status of each of the EXNET® ports. 0x00 Normal 0x01 Loop Back		
:	EXNET® Port B Status		

:	Master Status This field indicates whether or not this node is the master. 0x00 Not Master 0x01 Master
:	Transmit Mode This field indicates status of the transmit and receive modes. 0x00 Receive Only 0x01 Transmit/Receive 0xFF Transmit Mode Unassigned
:	Source Packet Number This field indicates the ID number of the source packet. 0x00–0x1F Source Packet Number 0xFF Source Packet Number Unassigned
:	Master Arbitration Flag This field indicates whether or not a node is allowed to become EXNET® Ring Master: 0x01 This node is allowed to participate in EXNET® Ring Master Arbitration. 0x00 This node is not allowed to participate in EXNET® Ring Master Arbitration.
:	Checksum

Redundancy State This field indicates redundancy state of the RICs:

- 0x00 No Redundancy
- 0x01 Active RIC
- 0x02 Standby RIC
- 0xFF Role Not Yet Defined

Current Ring Mode This field indicates the Ring Mode:

- 0x00 Normal Ring Mode
- 0x01 Enhanced Fault Tolerance Ring Mode

Route Control 0x00E8

SwitchKit Name Route Control

Type EXS API and SwitchKit API message

Description **Route Control 0x00E8**

Use this message to outseize to a channel using the CSP Call Control. If you want to the host application to choose the span/channel, use the *Outseize Control* message instead of this message.

Contact Dialogic Technical Support if you want to send a *Route Control* message greater than 512 bytes.

Sent by Host

Protocol-Specific Considerations

This section provides protocol-specific considerations to help you use this message for VoIP calls. Note that these are examples and that there are several other ways to use the *Route Control* message.

The mandatory TLVs are listed for each ICB, but other protocol specific TLVs are mandatory in order to initiate an outbound call. Moreover, you may include additional optional TLVs as needed.

SIP

For physical SIP calls, the *Route Control* message must contain the following:
AIB - 0x0029 - Generic Router

Extended ICBs

0x001E Generic PPL ICB

0x0013 Routing Method TLV
Value: 0x0008 Criteria Type

0x000F Router Protocol ID TLV
Value: 0x000B Default Router

0x0008 Criteria Type TLV
Value: 0x0065 Physical VoIP Channel

0x0065 Physical VoIP Channel TLV
Value: 0x0000 (reserved)

Route Control 0x00E8

0x0033 NPDI Universal ICB

0x277E Remote Side Protocol TLV

Value: 0x08 - SIP

H.323

For H.323 calls, the *Route Control* message must contain the following:

AIB - 0x0029 - Generic Router

Extended ICBs

0x001E Generic PPL ICB

0x0013 Routing Method TLV
Value: 0x0008 Criteria Type

0x000F Router Protocol ID TLV
Value: 0x000B Default Router

0x0008 Criteria Type TLV
Value: 0x0065 Physical VoIP Channel

0x0065 Physical VoIP Channel TLV
Value: 0x0000 (reserved)

0x0033 NPDI Universal ICB

0x277E Remote Side Protocol TLV
Value: 0x09H.323

0x27C2 Remote End Signaling IP Address (Q.931) TLV
Value: IP Address

SIP Call Agent

For Call Agent calls, the *Route Control* message must contain the following:

AIB - 0x0029 - Generic Router

Extended ICBs

0x001E Generic PPL ICB

0x0013 Routing Method TLV
Value: 0x0008 Criteria Type

0x000F Router Protocol ID TLV
Value: 0x000B Default Router

0x0008 Criteria Type TLV
Value: 0x0071 Virtual VoIP Channel

0x0071 Virtual VoIP Channel TLV
Value: 0x0000 (reserved)

0x0033 - NPDI Universal ICB

0x277E Remote Side Protocol TLV
Value: 0x08SIP

Interworking

For Interworking calls, the *Route Control* message must contain the following:

AIB - 0x0029 - Generic Router

Extended ICBs

0x001E Generic PPL ICB
0x0013 Routing Method TLV
Value: 0x0008 Criteria Type

0x000F Router Protocol ID TLV
Value: 0x000B Default Router

0x0008 Criteria Type TLV
Value: 0x0065 Physical VoIP Channel

0x0065 Physical VoIP Channel TLV
Value: 0x0000 (reserved)

0x0033 NPDI Universal ICB
0x277E Remote Side Protocol TLV
Value: 0x08SIP

Clear Channel VoIP

For Clear Channel VoIP calls, the *Route Control* message must contain the following:

ICBs

0x001E Generic PPL ICB or 0x1E Generic PPL ICB

0x0013 Routing Method TLV
Value: 0x0009 Terminating Channel

0x0009 Terminating Channel TLV

Refer to the *Developer's Guides* for examples that include the *Route Control* message.

SwitchKit Code C Structure

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE ICBCount;
    UBYTE Data[222];
} XL_RouteControl;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE Data[251];
} XL_RouteControlAck;
```

C++ Class

```
class XLC_RouteControl : public XLC_OneChannelOutbound {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getSpan() const;
    void setSpan(XBYTE x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    XBYTE getRouterHandle() const;
    void setRouterHandle(unsigned short x);
    UBYTE getICBCount() const;
    void setICBCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

C++ Class Response

```
class XLC_RouteControlAck : public XLC_AcknowledgeMessage
{
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00E8)	3, 4	Message Type (0x00E8)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB
:	Address Method	10	ICB
:	Number of AEs to follow	:	Checksum
:	AEs		
:	ICB		
:	:		
:	Checksum		

EXS API Hex Format - Detailed

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00E8)	3, 4	Message Type (0x00E8)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID

Route Control 0x00E8

8 :	<u>AIB</u> Address Method 0x00 - Individual AEs	8, 9	Status MSB, LSB
	Number of AEs to follow	:	Number of ICBs to follow
		:	ICB 0x02 Data ICBs 0x1E Generic PPL
		:	Number of TLVs to follow.
<u>AEs</u> <u>0x0D Channel</u> Do not use this AIB. You must use the Router AIB below: <u>0x29 Router</u> Using this AIB, the CSP seizes the outbound channel but does not connect. The host must establish the connection with a separate <i>Connect</i> message. This AIB indicates the RTR handle (always 0xFFFE), the CSP resolves the "B" channel using Route Tables and outseizes, then notifies the host, which connects. Use this AIB if you want the internal Router component to find a destination channel.	:	Only the following TLVs are included in this ICB in this response: 0x0009 Address Element TLV (For Expanded Span/Channel AIB) 0x2950 SIP Call ID (For call ID reporting of outbound calls.)	
	:	State If the preceding Status field is a positive acknowledgement, this field does not apply.	
:	Number of ICBs to follow	:	Checksum
:	ICB 0x01 Action ICBs 0x00 Null * 0x01 Scan For Wink N 0x02 Scan for ANI Request Off-Hook * 0x03 Scan for Dial Tone * 0x04 Report Call Processing Event 0x05 Outpulse Stage N Address Data 0x06 Wait for Host Address Data * 0x07 Wait For Host Control *		

Route Control 0x00E8

	<p>0x08 Send Host Acknowledgment * (the ISDN PRI card ignores the Send Host Acknowledgment)</p> <p>0x09 Do Call Progress Analysis (T-ONE calls only)</p> <p>0x0A Seize *</p> <p>0x0B Use Instruction List</p> <p>0x0C Reserved *</p> <p>0x0D Cancel R2 Receiver *</p> <p>0x0E Scan for Backward R2 Signal *</p> <p>0x0F Wait for Host Control with Answer Supervision *</p> <p>0x10 Do Call Progress Analysis Without Line Signaling</p> <p>0x11 Delay N Milliseconds</p> <p>* The ICB has no data, so you will always use 0x00 for the value of the Data Length field.</p> <p>0x02 Data ICBs</p> <p>0x01 Stage N Address Data</p> <p>0x02 Stage N Address Data</p> <p>0x10 ISDN Formatted IEs</p> <p>0x11 ISDN Raw IEs</p> <p>0x12 SS7 Parameters</p> <p>0x15 DASS2/DPNSS Raw Data</p> <p>0x1A Call Type</p> <p>0x1C SS7 TUP Formatted Fields</p> <p>0x1E Generic PPL</p> <p>0x25 ISDN Segmented Message</p> <p>0x03 Extended Data ICBs</p> <p>0x001E Generic PPL (H.323 / SIP)</p> <p>0x0026 Channel Pad Value</p> <p>0x0029 V5 Formatted IEs</p> <p>0x002C V5 User Port Range</p> <p>0x002F V5 Statistics Query</p> <p>0x0030 V5 Statistics Data</p> <p>0x0031 V5 Status Indication</p> <p>0x0033 NPDI Universal ICB (H.323 / SIP)</p>
	Refer to <i>Interworking TLVs (4-4)</i> if you have enabled interworking from the <i>VoIP Protocol Configure 0x00EE</i> message.
:	Checksum

ServerStatusChange

Type	SwitchKit API message
Description	The LLC sends the <i>ServerStatusChange</i> message containing all the information needed by an application to determine the current status of a node. The message is sent when the LLC discovers a change in CSP status.
C Structure	<pre>typedef struct { unsigned short Status; UBYTE SystemType; UBYTE MatrixSide; UBYTE MatrixState; UBYTE AdjMatrixState; UBYTE StatusBits; UBYTE Reserved1; UBYTE Reserved2; unsigned short ExcelPort; int PhysicalNode; UBYTE LogicalNode; UBYTE HostNode; UBYTE NodeStatus; UBYTE MessageTrigger; UBYTE FromPrimary; UBYTE SocketStatus; unsigned short UpdatedFieldBits; UBYTE reserved40[1]; } <i>SK_ServerStatusChange</i>;</pre>
C++ Class	<pre>class <i>SKC_ServerStatusChange</i> : public SKC_DummyMessage { public: unsigned short getStatus() const; void setStatus(unsigned short x); UBYTE getSystemType() const; void setSystemType(UBYTE x); UBYTE getMatrixSide() const; void setMatrixSide(UBYTE x); UBYTE getMatrixState() const; void setMatrixState(UBYTE x); UBYTE getAdjMatrixState() const; void setAdjMatrixState(UBYTE x); UBYTE getStatusBits() const; void setStatusBits(UBYTE x); UBYTE getReserved1() const; void setReserved1(UBYTE x); UBYTE getReserved2() const;</pre>

```

void setReserved2(UBYTE x);
unsigned short getExcelPort() const;
void setExcelPort(unsigned short x);
int getPhysicalNode() const;
void setPhysicalNode(int x);
UBYTE getLogicalNode() const;
void setLogicalNode(UBYTE x);
UBYTE getHostNode() const;
void setHostNode(UBYTE x);
UBYTE getNodeStatus() const;
void setNodeStatus(UBYTE x);
UBYTE getMessageTrigger() const;
void setMessageTrigger(UBYTE x);
UBYTE getFromPrimary() const;
void setFromPrimary(UBYTE x);
UBYTE getSocketStatus() const;
void setSocketStatus(UBYTE x);
unsigned short getUpdatedFieldBits();
void setUpdatedFieldBits(unsigned short x);
};

```

Conditions for Sending the ServerStatusChange Message

The following list shows the conditions that will cause the LLC to send a *ServerStatusChange* message:

- SK_PM_CHANGE - sent when the first poll is received by the LLC for a CSP Matrix Series 3 Card. Also sent any time a value changes in the Poll. The UpdatedFieldBits field indicates which fields have changed since the last poll. The following is a possible scenario:
 1. The adjacent matrix has lost connection to the LCC running on the host. Either the LAN connection is severed or the host has severe CPU deprivation. Diagnose the LAN connection. Determine if the matrix card can receive and return ping messages. If the LAN connection is good, check the performance of the host.
- SK_SSC_UPDATE - sent under the following conditions:
 1. When an application registers for the ServerStatusChange notification via *sk_msgRegister()*, an SSC_UPDATE is sent for each node the LLC is currently connected to which has a CSP Matrix Series 3 Card in the active state.
 2. When an application issues an *AddLLCNode*, an SSC_UPDATE may be generated if the LLC already knew about the node and the LLC is currently connected to the active CSP Matrix Series 3 Card for that node. In other words, if the Link is considered to be up.

- SK_LINK_UP - sent under the following conditions:
 1. This message is sent to indicate that there is a connection to the active CSP Matrix Series 3 Card for this node and the CSP Matrix Series 3 Card is ready to be configured. This will be sent if this is a new condition (i.e. we were not previously connected to the active CSP Matrix Series 3 Card).
 2. Sent if an application has performed an *AddLLCNode*, the LLC is currently connected to the specified node, and the LLC is connected to the active CSP Matrix Series 3 Card of that node.
 3. When an application registers for the *ServerStatusChange* notification via *sk_msgRegister()*, and the LLC is connected to the active CSP Matrix Series 3 Card of a node.
- SK_LINK_DOWN - sent under the following conditions:
 1. This message is sent to indicate that there is NO connection to the active CSP Matrix Series 3 Card for a node. This will be sent if this is a new condition (that is, we were previously connected to the active CSP Matrix Series 3 Card).
 2. Sent if an application has performed an *AddLLCNode*, and the LLC is not currently connected to the specified node's active CSP Matrix Series 3 Card.
 3. When an application registers for the *ServerStatusChange* notification via *sk_msgRegister()*, and the LLC is supposed to be connected to a node but is not connected to the node's active CSP Matrix Series 3 Card.

More on SK_LINK_UP and SK_LINK_DOWN

- Registering for *ServerStatusChange* will result in an SK_LINK_UP or SK_LINK_DOWN for each node controlled by the LLC. If an SK_LINK_UP is sent, it will be followed by an SK_SSC_UPDATE with the last poll from the active CSP Matrix Series 3 Card.
- Performing an *AddLLCNode* will result in an SK_LINK_UP or SK_LINK_DOWN for the specified node if the LLC is already controlling the node. If an SK_LINK_UP is sent, it will be followed by an SK_SSC_UPDATE with the last poll from the active CSP Matrix Series 3 Card, if the LLC is currently connected to the CSP Matrix Series 3 Card in the active state.

- SK_PM_CHANGE, SK_SOCKET_CHANGE, SK_LINK_UP and SK_LINK_DOWN are generated to indicate a change in state.
- SK_NSQ_CHANGE or SK_NSR_CHANGE are generated upon arrival of *NodeStatusQueryAck* or *NodeStatusReport* from the CSP.

Arguments

The following table indicates what arguments are valid for a specific message trigger:

MessageTrigger	Argument	Description
SK_PM_CHANGE or SK_SSC_UPDATE	PhysicalNode	Represents the requested Node ID
	LogicalNode	Represents the current Node ID as reported by the CSP in <i>PollMessage</i>
	SocketStatus	1 always. Socket must be connected for this indication to occur.
	Status	Status from <i>PollMessage</i>
	SystemType	System Type(Poll Source) from <i>PollMessage</i>
	MatrixSide	CSP Matrix Series 3 Card ID from <i>PollMessage</i>
	MatrixState	Matrix State(Card State) from <i>PollMessage</i>
	AdjMatrixState	Adjacent Card State from <i>PollMessage</i>
	StatusBits	Card Status from <i>PollMessage</i>
	CSP Port	Double Byte after Multi Host Connection Status Byte 2 in <i>PollMessage</i>
	UpdatedFieldBits	Bit mask indicating which fields in the poll were updated. See table below for details of the meaning of the bits. Only set for SK_PM_CHANGE
SK_LINK_UP or SK_LINK_DOWN	PhysicalNode	Requested Node ID
	LogicalNode	Requested Node ID

Constants The following table contains information for the field UpdatedFieldBits. You can combine any number of settings in this bit mask:

Constants	Changed Argument in <i>Server StatusChange</i> message
SK_STATUS_CHANGE	Status
SK_SYSTEMTYPE_CHANGE	SystemType
SK_MATRIXSIDE_CHANGE	MatrixSide
SK_MATRIXSTATE_CHANGE	MatrixState
SK_ADJMATRIXSTATE_CHANGE	AdjMatrixState
SK_STATUSBITS_CHANGE	StatusBits
SK_LOGICALNODE_CHANGE	LogicalNode

Service State Configure 0x000A

SwitchKit Name ServiceStateConfig

Type EXS API and SwitchKit API message

Description **Service State Configure 0x000A**

This message allows the host to bring an entity (a card, span, channel, range of channels, DSP SIMM, or SS7-controlled circuit) in and out of service. On the DSP Series 2 card, this message must use AIB 0x22 to take an individual DSP chip in and out of service.

A card is automatically brought into service when the system is powered up. The host does not have to bring a card into service unless it has previously taken the card out of service using this message.

DS3 Card

The DS3 framer is automatically brought into service when the DS3 card is brought into service. Once in service, DS3 status changes are reported to the host in the *Alarm* messages (0x00B9). If the DS3 circuit is not ready for operation, the host can disable DS3 alarms by placing the DS3 framer out of service. To do this, have the host send a *Service State Configure* message with the DS3 Offset AIB (0x32) and an action of 0x0F (Take Out of Service).

If the host does take the DS3 Framer Out of Service, it is the host's responsibility to bring it back in service. If the host does not bring the DS3 Framer back in service, the CSP will send a NACK 0x005A (DS3 Out of Service) when it receives the *Service State Configure* message to Brings Span In Service.

The DS3 Offset AIB must always be 0.

LAPD Super Rate

For LAPD Super Rate, this message must address the first D channel (DS0). The other D channels in the Super Rate D channel do not need to be addressed in *Service State Configure*.

Virtual Cards

You cannot use *Service State Configure* to take virtual cards out-of-service.

Alarm and Alarm Cleared messages

The CSP sends an *Alarm* message (0x00B9) to the host when the cards are taken out of service (OOS). When cards are brought back in service, the CSP sends an Alarm Cleared (0x00C1) message to the host.

For DSP modules, the CSP sends an Alarm Clear message to the host when the DSP Module is brought in service.

This functionality applies to all CSP cards except the following:

- I/O cards
- CSP Matrix Series 3 Card
- Virtual cards

The CSP sends information about the cards to the host in the *Card Status Report* message (0x00A6). Host applications act based on the Alarm and Alarm Cleared messages received from the CSP. This feature allows the application to track card status and take action without modifying the application.

Sent by Host

**Example Message
(Socket Log Output)**

The following example message (socket log output for SwitchKit) takes channels 0x01 through 0x05 on span 0x01 out of service.

```
00 13 00 0A 00 00 FF 01 02 0D 03 00 01 01 0D 03 00 01 05 0F 00
```

Configuration

```
ServiceStateConfig (
    Node = Integer,
    Range = StartSpan:StartChan - EndSpan:EndChan,
    Span = Integer,
    Slot = Integer,
    SIMM = Integer,
    StackID = Integer,
    LinkID = Integer,
    Ring = Integer,
    V5ID = Integer,
    DS3 = Integer,
    Action = Integer,
    ForcedFlag = Integer);
```

C Structure

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE Action;
    UBYTE ForcedFlag;
} XL_ServiceStateConfig;
```

C++ Class

```
class XLC_ServiceStateConfig : public
    XLC_ChanRangeMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getStartSpan() const;
    void setStartSpan(XBYTE x);
    UBYTE getStartChannel() const;
    void setStartChannel(UBYTE x);
    XBYTE getEndSpan() const;
    void setEndSpan(XBYTE x);
    UBYTE getEndChannel() const;
    void setEndChannel(UBYTE x);
    UBYTE getDSPChip() const;
    void setDSPChip(UBYTE x);
    UBYTE getDSPSlot() const;
```

```
void setDSPSlot(UBYTE x);
UBYTE getDSPSIMM() const;
void setDSPSIMM(UBYTE x);
UBYTE getSlot() const;
void setSlot(UBYTE x);
UBYTE getStackID() const;
void setStackID(UBYTE x);
UBYTE getLni() const;
void setLni(UBYTE x);
UBYTE getSIMM() const;
void setSIMM(UBYTE x);
UBYTE getDS3() const;
void setDS3(UBYTE x);
XBYTE getSpan() const;
void setSpan(XBYTE x);
XBYTE getV5ID() const;
void setV5ID(XBYTE x);
UBYTE getAction() const;
void setAction(UBYTE x);
UBYTE getForcedFlag() const;
void setForcedFlag(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x000A)	3, 4	Message Type (0x000A)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB) (See Invalid Responses)
:	Address Method 0x00 - Individual AEs or 0x01 - Range	10	Checksum
Number of AEs to follow			
AEs			
<u>Individual AEs</u>			
0x01 Slot			
0x09 SS7 Link			
0x0C Logical Span			
0x12 DSP SIMM			
0x1A EXNET Ring			
0x22 DSP Chip			
0x32 DS3 Offset (Always 0)			
0x48 Ring Communication Link			
0x2C V5 ID			
<u>Range AEs</u>			
0x0D Channel (Starting)			
0x0D Channel (Ending)			
:	Action		
0x0F Take Out of Service			
0xF0 Bring In Service			
:	Forced Flag		
Use this flag to determine the way a SIMM is taken out of service. The meaning of this field depends on the entity being addressed.			
0x04 DSP SIMM			
0x01 Take a DSP SIMM out of service, even if it is currently being used. When you force a DSP SIMM out of service, you receive no alarm.			
0x00 Take a DSP SIMM out of service gracefully. When you take a DSP SIMM out of service gracefully, you receive a DSP SIMM OOS Alarm.			
For all other entities: Reserved, must be 0x00.			
:	Checksum		

Invalid Responses

NOTE: These invalid responses apply to the V5 ID address type only

- 0x01-The host could be attempting to bring the V5 interface in service or out of service when the V5 Interface ID (4 bytes) has not been configured. The host could be attempting to bring the V5 interface in service or out of service when the V5 ID is not configured.
- 0x04- The host could be attempting to bring the V5 interface in service when the V5 Variant ID has not been configured for that interface.
- 0x07-The host could be attempting to bring the V5 interface in service when no user ports are configured for the V5 ID.
- 0x0A-The host could be attempting to bring the V5 interface in service when no C Channels are assigned to this V5 ID.
- 0x0B-The host was attempting to bring a V5 interface in service when a C Channel was found on an unconfigured link.
- 0x0DRESERVED
0x0ERESERVED
0x0FRESERVED

Notes:

- 1 When a DS3 is taken out of service, the CSP automatically takes all spans and channels on the DS3 out of service. When the DS3 is brought back in service, the spans and channels are *not* automatically brought back in service. You must manually bring the spans and channels in service. A DS3 must be in service before any spans can be brought in service. The CSP returns a status of *DS3 Out of Service* (0x5A) if you attempt to bring a span in service first.
- 2 When bringing a card in service that was previously out of service, the host must wait for a *Card Status Report* message on the appropriate slot before considering the card in service.
- 3 When a span is taken out of service, all channels on the span are automatically taken out of service by the CSP. When the span is brought back in service using this message, the channels are not automatically brought back in service. You must send this message again to bring the channels back into service.
- 4 The host should not consider a channel in service upon receiving the Response to this message. The host must

receive a *DSO Status Change* message with status of “In Service” before considering the channel in service.

- 5 If you are using E1 Channel 30 for purposes other than CAS (for example, CCS), you cannot include it in a range with other channels. Instead, you must bring E1 Channel 30 into service with a single separate *Service State Configure* message. Be sure that both the starting and ending spans and channels are set to the span and channel 30.
- 6 If you take an E1 PPL channel out of service using this message, the host receives the following:
 - A *DSO Status Change* message (0x42) that has a *Channel Status* field value of 0x01 (Out of Service).
 - A *DSO Status Change* message (0x42) that has a *Channel Status* field value of 0x00 (Purge) and a *Purge Reason* field value of 0x74 (Disabled While Non-dormant).The system sends this second message to the host as a result of meeting certain criteria but does not actually perform a purge. You can ignore this second message.

SNMP Configure 0x0092

SwitchKit Name	SNMPConfig
Type	EXS API and SwitchKit API message
Description	SNMP Configure 0x0092 This message uses the <i>Generic Report</i> message (0x46) to send statistical information to the CSP.
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {  
    UBYTE Slot;  
    UBYTE DS3;  
    UBYTE Type;  
    UBYTE Action;  
    UBYTE Period;  
} XL_SNMPCconfig;
```

C++ Class

```
class XLC_SNMPCconfig : public XLC_OutboundMessage {  
public:  
    UBYTE getSlot() const;  
    void setSlot(UBYTE x);  
    UBYTE getDS3() const;  
    void setDS3(UBYTE x);  
    UBYTE getType() const;  
    void setType(UBYTE x);  
    UBYTE getAction() const;  
    void setAction(UBYTE x);  
    UBYTE getPeriod() const;  
    void setPeriod(UBYTE x);  
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0092)	3, 4	Message Type (0x0092)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow	10	Checksum
	AE DS3 Offset: (0x32) 0x0C Logical Span		
:	Type This field instructs the CSP on how the report should be returned. 0x01 Return report using <i>Generic Report Message</i>		
:	Action This field contains a bit mask that indicates what actions are to be taken. If this field is 0x00, periodic report will be turned off. Bit 0: Clear counters if set Bit 1: Send a report only once Bit 2: Send report periodically		
:	Period This field contains the interval in minutes in which a periodic report should be sent. If this field is 0x00, it will be set to 0x0F (15 minutes).		
:	Checksum		

AIB This AIB addresses a DS3 offset for a DS3 card. It is used to bring a DS3 in service and out of service:

Byte	AIB Field Description
2	Address Type (0x32)
3	Data Length (0x02)
4	Data[0] – Slot Number
5	Data[1] DS3 Offset

Span Filter Configure 0x00CD

SwitchKit Name	Span Filter Config
Type	EXS API and SwitchKit API message
Description	<p>Span Filter Configure 0x00CD</p> <p>This message is used to configure the detection of a Carrier Group Alarm (CGA), such as an AIS (Alarm Indication Signal) or LOF (Loss of Frame). The “CGA Detected” validation timer determines how long a CGA alarm must be detected before the CSP informs the host with an <i>Alarm</i> message. The default is 2.5 seconds. The host will also receive a <i>DSO Status Change</i> message of “Out Of Service” for the channels on the span.</p> <p>The “CGA Cleared” validation timer is initiated after a CGA alarm has been detected and then clears. It determines how long the CSP must go without detecting a CGA alarm before informing the host with an <i>Alarm Cleared</i> message. The default is 12 seconds. The host will also receive a <i>DSO Status Change</i> message of “In Service” for the channels on the span.</p>
Sent by	Host
SwitchKit Code	<p>Configuration</p> <pre>SpanFilterConfig (Node = integer, Span = integer, Entity = integer, Value = integer);</pre> <p>C Structure</p> <pre>typedef struct { unsigned short Span; UBYTE Entity; unsigned short Value; } XL_SpanFilterConfig;</pre> <p>C++ Class</p> <pre>class XLC_SpanFilterConfig : public XLC_SpanMessage { public: unsigned short getSpan() const; void setSpan(unsigned short x); UBYTE getEntity() const; void setEntity(UBYTE x); unsigned short getValue() const; void setValue(unsigned short x); };</pre>

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00CD)	3, 4	Message Type (0x00CD)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB
:	Address Method 0x00 - Individual AEs	10	Checksum
	Number of AEs to follow		
	AEs 0x0C Logical Span		
:	Entity 0x01 "CGA Cleared" validation timer 0x02 "CGA Detected" validation timer		
:	Value MSB, LSB Must be in 10 ms units.		
:	Checksum		

Span Filter Query 0x00CE

SwitchKit Name	SpanFilterQuery
Type	EXS API and SwitchKit API message
Description	<p>Span Filter Query 0x00CE</p> <p>This message is used to query the values of the CGA detected and cleared validation timers</p> <p>The “CGA Detected” validation timer determines how long a CGA alarm must be detected before the CSP informs the host with an <i>Alarm</i> message. The default is 2.5 seconds. The host will also receive a <i>DSO Status Change</i> message of “Out Of Service” for the channels on the span.</p> <p>The “CGA Cleared” validation timer is initiated after a CGA alarm has been detected and then clears. It determines how long the CSP must go without detecting a CGA alarm before informing the host with an <i>Alarm Cleared</i> message. The default is 12 seconds. The host will also receive a <i>DSO Status Change</i> message of “In Service” for the channels on the span.</p>
Sent by	Host
SwitchKit Code	<p>C Structure</p> <pre>typedef struct { unsigned short Span; } XL_SpanFilterQuery;</pre> <p>C Structure Response</p> <pre>typedef struct { unsigned short Status; unsigned short NoCGAFilter; unsigned short CGAFilter; } XL_SpanFilterQueryAck;</pre> <p>C++ Class</p> <pre>class XLC_SpanFilterQuery : public XLC_OutboundMessage { public: unsigned short getSpan() const; void setSpan(unsigned short x); };</pre> <p>C++ Class Response</p> <pre>class XLC_SpanFilterQueryAck : public XLC_AcknowledgeMessage { public: unsigned short getStatus() const; void setStatus(unsigned short x); unsigned short getNoCGAFilter() const;</pre>

Span Filter Query 0x00CE

```
void setNoCGAFilter(unsigned short x);
unsigned short getCGAFilter() const;
void setCGAFilter(unsigned short x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x000B)
3, 4	Message Type (0x00CE)	3, 4	Message Type (0x00CE)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8, 9	Status MSB, LSB
:	Address Method 0x00 - Individual AEs	10, 11	No CGA Filter MSB, LSB
	Number of AEs to follow	12, 13	CGA Filter MSB, LSB
	AEs 0x0C Logical Span	14	Checksum
:	Checksum		

Span Status Query 0x0084

SwitchKit Name	SpanStatusQuery
Type	EXS API and SwitchKit API message
Description	<p>Span Status Query 0x0084</p> <p>This message may be used by the host to obtain information about the state of an in-service T1 or E1 span. It resets bipolar violations and frame error contents.</p> <p>For the SS7 Multi-Protocol I/O card, the positive acknowledgement to this query contains zeros in the data fields.</p>
Sent by	Host

SwitchKit Code C Structure

```
typedef struct {
    unsigned short Status;
    unsigned short Span;
    UBYTE AlarmStatus;
    unsigned short BiPolarCntr;
    unsigned short FrameErrors;
    unsigned short SlipCntr;
    unsigned short RemoteAlarmCntr;
    unsigned short FrameLossCntr;
} XL_SpanStatusQueryAck;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    unsigned short Span;
    UBYTE AlarmStatus;
    unsigned short BiPolarCntr;
    unsigned short FrameErrors;
    unsigned short SlipCntr;
    unsigned short RemoteAlarmCntr;
    UBYTE FrameLossCntr;
} XL_SpanStatusQueryAck;
```

C++ Class

```
class XLC_SpanStatusQuery : public XLC_OutboundMessage {
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
};
```

C++ Class Response

```

class XLC_SpanStatusQueryAck : public
  XLC_AcknowledgeMessage {
  public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getAlarmStatus() const;
    void setAlarmStatus(UBYTE x);
    unsigned short getBiPolarCntr() const;
    void setBiPolarCntr(unsigned short x);
    unsigned short getFrameErrors() const;
    void setFrameErrors(unsigned short x);
    unsigned short getSlipCntr() const;
    void setSlipCntr(unsigned short x);
    unsigned short getRemoteAlarmCntr() const;
    void setRemoteAlarmCntr(unsigned short x);
    unsigned short getFrameLossCntr() const;
    void setFrameLossCntr(unsigned short x);
  };

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x0084)	3, 4	Message Type (0x0084)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB If this field is anything other than 0x0010 (Positive Acknowledgment), the value of the Length field will be 0x0003.
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow	10	AIB (same as message)
	AE 0x0C Logical Span		
:	Checksum		
Response continued below.			

Span Status Query 0x0084

:	<p>Alarm Status</p> <p>This field is a bit mask. The bit values are 0=Disabled, 1=Enabled.</p> <ul style="list-style-type: none"> Bit 0 Receiving Red Alarm Bit 1 Receiving Yellow Alarm Bit 2 Receiving Loss of Signal Bit 3 Receiving Out of Frame Bit 4 Sending Red Alarm Bit 5 Sending Yellow Alarm Bit 6 Receiving AIS Bit 7 In Remote Loopback
:	<p>Bipolar Violation Counter MSB, LSB</p> <p>The number of bipolar violations detected since last request.</p>
:	<p>Frame Errors MSB, LSB</p> <p>If the span is T1, this field contains the number of frame bit errors. If the span is E1, this field contains frame alignment signal errors.</p>
:	<p>Slip Counter MSB, LSB</p> <p>The number of slips detected since last request.</p>
:	<p>Remote Alarm Indication Counter MSB, LSB</p> <p>The number of times Remote Alarm Indication has been detected since the last request.</p>
:	<p>Loss of Frame Counter MSB, LSB</p> <p>The Number of times a Loss of Frame has been detected since the last request.</p>
:	<p>Checksum</p>

SS7 CIC Configure 0x006A

SwitchKit Name	SS7CICConfig
Type	EXS API and SwitchKit API message
Description	<p>SS7 CIC Configure 0x006A</p> <p>This message allows the host to define an association between channels on logical spans and SS7 CICs. CICs in a CIC group must be on the same span and in contiguous timeslots. You can also use this message to deconfigure CIC group.</p> <p>You must use this message for one CIC group at a time. You cannot use this message on multiple CIC groups.</p> <p>To deconfigure a CIC group, use this message but substitute the value 0xFFFFFFFF for the DPC. If the CICs are in-service, the host receives DS0 status messages for the CICs with a status of out-of-service.</p>
Sent by	Host
SwitchKit Code	<p>Configuration</p> <pre>SS7CICConfig (Node = integer, StackID = integer, BaseCICNumber = integer, BaseCICSpan = integer, BaseCICChannel = integer, NumCICs = integer, DPC = integer, CallControlUserPart = integer);</pre> <p>C Structure</p> <pre>typedef struct { UBYTE StackID; unsigned short BaseCICNumber; unsigned short BaseCICSpan; UBYTE BaseCICChannel; UBYTE NumCICs; int DPC; UBYTE CallControlUserPart; } XL_SS7CICConfig;</pre> <p>C++ Class</p> <pre>class XLC_SS7CICConfig : public XLC_OutboundMessage { public: UBYTE getStackID() const; void setStackID(UBYTE x); unsigned short getBaseCICNumber() const; void setBaseCICNumber(unsigned short x); unsigned short getBaseCICSpan() const;</pre>

```

void setBaseCICSpan(unsigned short x);
UBYTE getBaseCICChannel() const;
void setBaseCICChannel(UBYTE x);
UBYTE getNumCICs() const;
void setNumCICs(UBYTE x);
int getDPC() const;
void setDPC(int x);
UBYTE getCallControlUserPart() const;
void setCallControlUserPart(UBYTE x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x006A)	3, 4	Message Type (0x006A)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	10	Checksum
:	Address Method 0x00 - Individual AEs or 0x01 - Range		
	Number of AEs to follow	8, 9	Status MSB, LSB
	AEs For Voice Circuit: 0x14 SS7 CIC Group For Virtual CIC: 0x31 SS7 Virtual CIC (can be used with Individual AEs or Range address method)		
:	DPC (four bytes) If using the BT IUP protocol, the maximum range for a DPC is 14 bits.		
:	Call Control User Part This field indicates the Call Control User Part for the CIC: 0x00 ISUP (Default) 0x01 TUP 0x02 ISUP/Japanese ISUP CIC Mapping (See information below this table) 0x31 SS7 VCIC		
:	Checksum		

CIC Assignment over E1

For CIC assignments over E1, the base channel refers to the Dialogic channel shown in the table below. The number of CICs defines the CIC range using the contiguous timeslots as the range. In the example table below there is no signaling link on the span. There is a base CIC of 0x00 and 31 CICs that can be configured as one group. If there is a signaling link on the span on a

timeslot other than timeslot 1 (Dialogic channel 0x00) then the CIC configure has to be split in two groups.

Note that this table is an example and reflects one way to assign CICs over E1.

Dialogic Channel	Timeslot	Full-span No SS7 Link	Last Channel Used as SS7 Link	First Channel Used as SS7 Link
31	0	-	-	-
0	1	0x00	0x00	SS7 Link
1	2	0x01	0x01	0x00
2	3	0x02	0x02	0x01
3	4	0x03	0x03	0x02
4	5	0x04	0x04	0x03
5	6	0x05	0x05	0x04
6	7	0x06	0x06	0x05
7	8	0x07	0x07	0x06
8	9	0x08	0x08	0x07
9	10	0x09	0x09	0x08
10	11	0x0A	0x0A	0x09
11	12	0x0B	0x0B	0x0A
12	13	0x0C	0x0C	0x0B
13	14	0x0D	0x0D	0x0C
14	15	0x0E	0x0E	0x0D
30	16	0x0F	SS7 Link	0x0E
15	17	0x10	0x0F	0x0F
16	18	0x11	0x10	0x10
17	19	0x12	0x11	0x11
18	20	0x13	0x12	0x12
19	21	0x14	0x13	0x13
20	22	0x15	0x14	0x14
21	23	0x16	0x15	0x15
22	24	0x17	0x16	0x16
23	25	0x18	0x17	0x17
24	26	0x19	0x18	0x18
25	27	0x1A	0x19	0x19
26	28	0x1B	0x1A	0x1A
27	29	0x1C	0x1B	0x1B
28	30	0x1D	0x1C	0x1C
29	31	0x1E	0x1D	0x1D

Call Control User Part

Japanese ISUP CIC Mapping Options

If you select ISUP/Japanese ISUP CIC Mapping (0x02), you must map CICs according to one of the following three options:

- Full span
- Last channel not used as CIC

- First channel not used as CIC

The values of the channels based for each option above are shown in the table below.

CSP Channel	Timeslot	Full-span No SS7 Link	Last Channel Used as SS7 Link	First Channel Used as SS7 Link
31	0	Frame Alignment Signal		
0	1	Base CIC + 0	Base CIC + 0	Not used as CIC (may be SS7 link)
1	2	Base CIC + 6	Base CIC + 6	Base CIC + 5
2	3	Base CIC + 12	Base CIC + 12	Base CIC + 11
3	4	Base CIC + 18	Base CIC + 18	Base CIC + 17
4	5	Base CIC + 24	Base CIC + 24	Base CIC + 23
5	6	Base CIC + 1	Base CIC + 1	Base CIC + 0
6	7	Base CIC + 7	Base CIC + 7	Base CIC + 6
7	8	Base CIC + 13	Base CIC + 13	Base CIC + 12
8	9	Base CIC + 19	Base CIC + 19	Base CIC + 18
9	10	Base CIC + 25	Base CIC + 25	Base CIC + 24
10	11	Base CIC + 2	Base CIC + 2	Base CIC + 1
11	12	Base CIC + 8	Base CIC + 8	Base CIC + 7
12	13	Base CIC + 14	Base CIC + 14	Base CIC + 13
13	14	Base CIC + 20	Base CIC + 20	Base CIC + 19
14	15	Base CIC + 26	Base CIC + 26	Base CIC + 25
30	16	Cannot Be Used		
15	17	Base CIC + 3	Base CIC + 3	Base CIC + 2
16	18	Base CIC + 9	Base CIC + 9	Base CIC + 8
17	19	Base CIC + 15	Base CIC + 15	Base CIC + 14
18	20	Base CIC + 21	Base CIC + 21	Base CIC + 20
19	21	Base CIC + 27	Base CIC + 27	Base CIC + 26
20	22	Base CIC + 4	Base CIC + 4	Base CIC + 3
21	23	Base CIC + 10	Base CIC + 10	Base CIC + 9
22	24	Base CIC + 16	Base CIC + 16	Base CIC + 15
23	25	Base CIC + 22	Base CIC + 22	Base CIC + 21
24	26	Base CIC + 28	Base CIC + 28	Base CIC + 27
25	27	Base CIC + 5	Base CIC + 5	Base CIC + 4
26	28	Base CIC + 11	Base CIC + 11	Base CIC + 10
27	29	Base CIC + 17	Base CIC + 17	Base CIC + 16
28	30	Base CIC + 23	Base CIC + 23	Base CIC + 22
29	31	Base CIC + 29	Not used as CIC (may be SS7 link)	Base CIC + 28

Response Data

The meaning of these fields depends upon the value of the *Entity* field in the message. The following value can be received:

- 0x7F Software module still locked.
This value is returned when attempting to configure a locked module for which no product license has been downloaded.

0xFD Node ID Not Configured

This response is a result of an SS7 server node configuration. Assignment of spans failed because node is not yet assigned. When sending the *SS7 CIC Configure* message (0x006A) over the EXNET® ring, this response value indicates the Ethernet connection is not available to the SS7 I/O card and you should check the connection.

SS7 CIC Query 0x0067

SwitchKit Name	SS7CICQuery
Type	EXS API and SwitchKit API message
Description	SS7 CIC Query 0x0067 This message retrieves CIC configuration information.
Sent by	Host
SwitchKit Code	C Structure

```
typedef struct {
    UBYTE StackID;
    int DPC;
    unsigned short BaseCIC;
    UBYTE NumCICs;
    UBYTE CallControlUserPart;
} XL_SS7CICQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE StackID;
    unsigned short BaseCICNumber;
    unsigned short BaseCICSpan;
    UBYTE BaseCICChannel;
    UBYTE NumCICs;
    int DPC;
    UBYTE Data[217];
} XL_SS7CICQueryAck;
```

C++ Class

```
class XLC_SS7CICQuery : public XLC_OutboundMessage {
public:
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    int getDPC() const;
    void setDPC(int x);
    unsigned short getBaseCIC() const;
    void setBaseCIC(unsigned short x);
    UBYTE getNumCICs() const;
    void setNumCICs(UBYTE x);
    UBYTE getCallControlUserPart() const;
    void setCallControlUserPart(UBYTE x);
};
```

C++ Class Response

```
class XLC_SS7CICQueryAck : public XLC_AcknowledgeMessage
{
public:
```

```

unsigned short getStatus() const;
void setStatus(unsigned short x);
UBYTE getStackID() const;
void setStackID(UBYTE x);
unsigned short getBaseCICNumber() const;
void setBaseCICNumber(unsigned short x);
unsigned short getBaseCICSpan() const;
void setBaseCICSpan(unsigned short x);
UBYTE getBaseCICChannel() const;
void setBaseCICChannel(UBYTE x);
UBYTE getNumCICs() const;
void setNumCICs(UBYTE x);
int getDPC() const ;
void setDPC(int x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};
    
```

EXS API Hex Format

NOTE: The five fields listed in the Response (Gray) starting with (CIC n Status and ending with CIC n PCM Encoding Format) are repeated for each CIC. The SS7 CIC Group address type contains the number of CICs.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1	Length (0xNNNN)	1	Length (0xNNNN)
3	Message Type (0x0067)	3	Message Type (0x0067)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u> : 0x00 - Individual AEs or 0x01 - Range	8, 9	Status MSB, LSB
	Number of AEs to follow	10	AIB : Address Method 0x00 - Individual AEs
	AEs Individual AEs 0x31 SS7 Virtual CIC 0x08 SS7 Stack or Range 0x31 SS7 Virtual CIC		Number of AEs to follow AE 0x14 SS7 CIC Group 0x08 SS7 Stack

:	DPC (four bytes) The configured Destination Point Code value for the selected Destination ID.	:	DPC (four bytes) for CIC Group 1 Base CIC for CIC Group 1 CIC counts for CIC Group 1 ... DPC (four bytes) for CIC Group N Base CIC for CIC Group N CIC counts for CIC Group N																
:	Base CIC Number MSB, LSB																		
:	Number of CICs in Query The number of CICs cannot exceed the number of CICs defined for a CIC group. Only one CIC group can be queried at one time.																		
:	Call Control User Part This field indicates the Call Control User Part used for the CIC: 0x00 ISUP 0x01 TUP 0x02 ISUP/Japanese ISUP CIC Mapping 0x31 SS7 VCIC																		
:	Checksum																		
:	Response continued below. CIC n Status (MSB, LSB only if using TUP) This field is a bit mask. The length of this field depends on the value of the <i>Call Control User Part</i> field: If the value of the <i>Call Control User Part</i> field is 0x00 or 0x02 (ISUP), this field is 1 byte. If the value of the <i>Call Control User Part</i> field is 0x01 (TUP), the field is 2 bytes. ISUP (1 Byte) <table style="margin-left: 20px;"> <thead> <tr> <th style="text-align: left;">Bit</th> <th style="text-align: left;">Status</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Circuit Remote Hardware Block State (ITU only)</td> </tr> <tr> <td>1</td> <td>Circuit Local Hardware Block State (ITU only)</td> </tr> <tr> <td>2</td> <td>Circuit Remote Maintenance Block State</td> </tr> <tr> <td>3</td> <td>Circuit Local Maintenance Block State</td> </tr> </tbody> </table> <p style="margin-left: 20px;">4–5 CPC State</p> <table style="margin-left: 20px;"> <tbody> <tr> <td>0</td> <td>Idle</td> </tr> <tr> <td>1</td> <td>Incoming Busy</td> </tr> <tr> <td>2</td> <td>Outgoing Busy</td> </tr> </tbody> </table> <p style="margin-left: 20px;">6 Circuit Transient State 7 Circuit Locally Unequipped State</p>			Bit	Status	0	Circuit Remote Hardware Block State (ITU only)	1	Circuit Local Hardware Block State (ITU only)	2	Circuit Remote Maintenance Block State	3	Circuit Local Maintenance Block State	0	Idle	1	Incoming Busy	2	Outgoing Busy
Bit	Status																		
0	Circuit Remote Hardware Block State (ITU only)																		
1	Circuit Local Hardware Block State (ITU only)																		
2	Circuit Remote Maintenance Block State																		
3	Circuit Local Maintenance Block State																		
0	Idle																		
1	Incoming Busy																		
2	Outgoing Busy																		

TUP (2 Bytes)

Bit Status

0 Remote Maintenance Block Status

0 Not Blocked

1 Blocked

1 Local Maintenance Block Status

0 Not Blocked

1 Blocked

2 Remote Hardware Block Status

0 Not Blocked

1 Blocked

3 Local Hardware Block Status

0 Not Blocked

1 Blocked

4 Remote Software Block Status

0 Not Blocked

1 Blocked

5 Local Software Block Status

0 Not Blocked

1 Blocked

6–7 Reserved

8–10 Call Status

0 Unknown

1 Idle

2 Incoming Seized

3 Outgoing Seized

4 Incoming CC Active

11 Equipped Status

0 Not Equipped

1 Equipped

12–15 Reserved

SSUTR2 (2 Bytes)	
<u>Bit</u>	<u>Status</u>
0	Outgoing Maintenance Block Status
	0 Not Blocked
	1 Blocked
1	Incoming Maintenance Block Status
	0 Not Blocked
	1 Blocked
2	Outgoing Manual Block Status
	0 Not Blocked
	1 Blocked
3	Transmission Error Block Status
	0 Not Blocked
	1 Blocked
4	Internal System Error Block Status
	0 Not Blocked
	1 Blocked
5	Reset Active Status
	0 Not Active
	1 Active
6-7	Test Call Status
	0 Idle
	1 Outgoing Call
	2 Incoming Call
8-9	Call Status
	0 Idle
	1 Outgoing Call
	2 Incoming Call

:	10 Incoming Community status 0 Not Active 1 Active 11 Outgoing Continuity Status 0 Not Active 1 Active 12 Equipped Status 0 Not equipped 1 Equipped 13-15 Reserved
	CIC n Local End Release Mode 0x01 Park 0x02 Release (Default)
:	CIC n Distant End Release Mode (not applicable for SS7 VCIC): 0x01 Park 0x02 Release (Default)
:	CIC n Answer Supervision (not applicable for SS7 VCIC): 0x00 Propagate Answer to the Distant End (Default) 0x01 Notify Host of Answer 0x02 Propagate Answer to the Distant End and Notify Host of Answer 0x03 No Answer Supervision (No propagation notification of Answer)
:	CIC n PCM Encoding Format (not applicable for SS7 VCIC): 0x01 μ -law Encoding (default for ANSI ISUP) 0x02 A-law Encoding (default for ITU ISUP)
:	Checksum

CIC Status SS7 Virtual CIC
 0x01 Virtual CIC
 0x00 Physical CIC

SS7 CLLI Configure 0x00EA

SwitchKit Name	SS7CLLIConfig
Type	EXS API and SwitchKit API message
Description:	SS7 CLLI Configure 0x00EA This message allows the host to configure CLLI (Common Language Location Identifier) information.
Sent by:	Host
SwitchKit Code	Configuration

```
SS7CLLIConfig (
    Node = integer,
    StackID = integer,
    BaseCICNumber = integer,
    BaseCICSpan = integer,
    BaseCICChannel = integer,
    NumCICs = integer,
    DataFormat = integer,
    TLVCount = integer,
    TLVData= byte array);
```

C Structure

```
typedef struct {
    UBYTE StackID;
    unsigned short BaseCICNumber;
    unsigned short BaseCICSpan;
    UBYTE BaseCICChannel;
    UBYTE NumCICs;
    UBYTE DataFormat;
    UBYTE TLVCount;
    UBYTE TLVData[221];
} XL_SS7CLLIConfig;
```

C++ Class

```
class XLC_SS7CLLIConfig : public XLC_OutboundMessage {
public:
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    unsigned short getBaseCICNumber() const;
    void setBaseCICNumber(unsigned short x);
    unsigned short getBaseCICSpan() const;
    void setBaseCICSpan(unsigned short x);
    UBYTE getBaseCICChannel() const;
    void setBaseCICChannel(UBYTE x);
    UBYTE getNumCICs() const;
    void setNumCICs(UBYTE x);
    UBYTE getDataFormat() const;
    void setDataFormat(UBYTE x);
```

```

UBYTE getTLVCount() const;
void setTLVCount(UBYTE x);
const UBYTE *getTLVData() const;
UBYTE *getTLVData();
void setTLVData(UBYTE *x);
};
    
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00EA)	3, 4	Message Type (0x00EA)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status, MSB, LSB
:	Address MEthod 0x00 - Individual AEs	10	Checksum
	Number of AEs to follow		
	AEs 0x14 SS7 CIC Group		
:	Data Format 0x01 TLVs		
	Number of TLVs 0x00-0x04		
:	Data 0x01 Trunk Number 0x02 Local CLLI 0x03 Remote CLLI 0x04 Trunk Type		
:	Checksum		

SS7 CLLI Query 0x00EB

SwitchKit Name	SS7CLLIQuery
Type	EXS API and SwitchKit API message
Description:	SS7 CLLI Query 0x00EB This message allows the host to query CLLI (Common Language Location Identifier) information.
Sent by:	Host
SwitchKit Code	C Structure

```
typedef struct {
    UBYTE StackID;
    unsigned short BaseCICNumber;
    unsigned short BaseCICSpan;
    UBYTE BaseCICChannel;
    UBYTE NumCICs;
} XL_SS7CLLIQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE StackID;
    unsigned short BaseCICNumber;
    unsigned short BaseCICSpan;
    UBYTE BaseCICChannel;
    UBYTE NumCICs;
    UBYTE DataFormat;
    UBYTE TLVCount;
    UBYTE TLVData[219];
} XL_SS7CLLIQueryAck;
```

C++ Class

```
class XLC_SS7CLLIQuery : public XLC_OutboundMessage {
public:
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    unsigned short getBaseCICNumber() const ;
    void setBaseCICNumber(unsigned short x);
    unsigned short getBaseCICSpan() const;
    void setBaseCICSpan(unsigned short x);
    UBYTE getBaseCICChannel() const ;
    void setBaseCICChannel(UBYTE x);
    UBYTE getNumCICs() const;
    void setNumCICs(UBYTE x);
};
```

C++ Class Response

```

class XLC_SS7CLLIQueryAck : public XLC_AcknowledgeMessage
{
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    unsigned short getBaseCICNumber() const;
    void setBaseCICNumber(unsigned short x);
    unsigned short getBaseCICSpan() const;
    void setBaseCICSpan(unsigned short x);
    UBYTE getBaseCICChannel() const;
    void setBaseCICChannel(UBYTE x);
    UBYTE getNumCICs() const;
    void setNumCICs(UBYTE x);
    UBYTE getDataFormat() const;
    void setDataFormat(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getTLVData() const;
    UBYTE *getTLVData();
    void setTLVData(UBYTE *x);
};
    
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00EB)	3, 4	Message Type (0x00EB)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u> Address Method 0x00 - Individual AEs	8, 9	Status MSB, LSB
	Number of AEs to follow	:	AIB Address Method 0x00 - Individual AEs
	AEs 0x14 SS7 CIC Group		Number of AEs to follow
:	Checksum		AEs 0x14 SS7 CIC Group

SS7 CLLI Query 0x00EB

:	Data Format
	0x01 (TLVs)
	TLV Count 0x00-0x04
	Data
:	0x01 Trunk Number
:	0x02 Local CLLI
:	0x03 Remote CLLI
:	0x04 Trunk Type
	Checksum

SS7 ISUP Message Format Configure 0x006B

SwitchKit Name	SS7ISUPMessageFormatConfig
Type	EXS API and SwitchKit API message
Description	<p>SS7 ISUP Message Format Configure 0x006B</p> <p>Use this message to add, remove or modify the format of an ISUP message in the ISUP Message Format table.</p> <p>Each of these configuration messages is used to format outgoing ISUP messages and to interpret incoming ISUP messages.</p> <p>For each message, there can be a maximum of eight Mandatory Fixed Parameters, eight Mandatory Variable Parameters, and 42 Optional Parameters.</p> <p>Use the <i>SS7 ISUP Message Query</i> message to retrieve the currently configured format of an ISUP message.</p> <p>Modifying the format of an ISUP message may also require modification of its PPL Configuration Bytes.</p>

Sent by Host

SwitchKit Code **Configuration**

```
SS7ISUPMessageFormatConfig (
    Node = integer,
    StackID = integer,
    ISUPMessageIndex = integer,
    ISUPMessageID = integer,
    ISUPMessagePriority = integer,
    NumMFPS = integer,
    Data = byte array);
```

C Structure

```
typedef struct {
    UBYTE StackID;
    UBYTE ISUPMessageIndex;
    UBYTE ISUPMessageID;
    UBYTE ISUPMessagePriority;
    UBYTE NumMFPS;
    UBYTE Data[219];
} XL_SS7ISUPMessageFormatConfig;
```

C++ Class

```
class XLC_SS7ISUPMessageFormatConfig : public
    XLC_OutboundMessage {
public:
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    UBYTE getISUPMessageIndex() const;
    void setISUPMessageIndex(UBYTE x);
```

```

UBYTE getISUPMessageID() const;
void setISUPMessageID(UBYTE x);
UBYTE getISUPMessagePriority() const;
void setISUPMessagePriority(UBYTE x);
UBYTE getNumMFPS() const;
void setNumMFPS(UBYTE x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x006B)	3, 4	Message Type (0x006B)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB
:	Address MEthod	10	Checksum
	0x00 - Individual AEs		
	Number of AEs to follow		
	AEs		
	0x08 SS7 Stack		
:	ISUP Message Index		
	See information below this table		
:	ISUP Message ID		
	See information below this table		
:	ISUP Message Priority		
	This field applies for ANSI only (for ITU enter 0x00). See T1.111.5 Appendix B for ANSI values.		
:	Number of Mandatory Fixed Parameters (MFP)		
:	MFP 1: ID		
:	MFP 1: Length		
:	MFP n: ID		
:	MFP n: Length		
:	Number of Mandatory Variable Parameters (MVP)		
:	MVP 1: ID		
:	MVP 1: Minimum Length		
:	MVP 1: Maximum Length (See Maximum Length at the end of this message description.)		
:	MVP n: ID		
:	MVP n: Minimum Length		
:	MVP n: Maximum Length (See Maximum Length at the end of this message description.)		
:	Optional Parameters Allowed (See Repeatable Optional Parameters/Optional Parameters Allowed at the end of this message description.)		

SS7 ISUP Message Format Configure 0x006B

:	Number of Optional Parameters
:	OP 1: ID
:	OP 1: Minimum Length
:	OP 1: Maximum Length (See Maximum Length at the end of this message description.)
:	OP n: ID
:	OP n: Minimum Length
:	OP n: Maximum Length (See Maximum Length at the end of this message description.)
:	Repeatable Optional Parameters
:	ROP 1: ID
:	ROP 1: Minimum Length
:	ROP 1: Maximum Length (See Maximum Length at the end of this message description.)
:	ROP n: ID
:	ROP n: Minimum Length
:	ROP n: Maximum Length (See Maximum Length at the end of this message description.)
:	Repeatable Optional Parameters
:	ROP 1: ID ROP n: ID
:	Number of TLVs
:	TLV Both TLVs must be sent in the message and must be set to the same value. To remove an ISUP message, set both TLVs to 0x00. To add an ISUP message, set both TLVs to 0x01. 0x4E21 Message Configuration Valid 0x4E22 Message Processing Supported
:	Checksum

Additional AIB

:	ISUP Message Index
:	Turnon/Turnoff Tag, MSB (0x00)
:	Turnon/Turnoff Tag, LSB (0xff)
:	# of TLVs attached
:	TLV 1: Tag, MSB
:	TLV 1: Tag, LSB
:	TLV 1: Length, MSB
:	TLV 1: Length, LSB
:	TLV 1: Data [0]
:	:
:	TLV 1: Data [n]
:	TLV n: Tag, MSB
:	TLV n: Tag, LSB
:	TLV n: Length, MSB
:	TLV n: Length, LSB
:	TLV n: Data[0]
:	:
:	TLV n: Data[n]

ISUP Message Index

This field specifies an index that is maintained for internal use by the CSP into the ISUP Message Format table for the specified ISUP message. The valid entries for this field are in numeric order.

Message	Index		Message	Index		Message	Index
IAM	0x00		LPA	0x14		CVT	0x28
ACM	0x01		CFN	0x15		CVR	0x29
ANM	0x02		UCIC	0x16		CRM	0x2A
CON	0x03		CQM	0x17		CRA	0x2B
REL	0x04		CQR	0x18		UPT	0x2C
RLC	0x05		FAR	0x19		UPA	0x2D
CPG	0x06		FAA	0x1A		EXM	0x2E
BLO	0x07		FRJ	0x1B		MPM	0x4F
BLA	0x08		USR	0x1C		OPR	0x50
UBL	0x09		FAC	0x1D		CCL	0x51
UBA	0x0A		PAM	0x1E		DRS	0x52
RSC	0x0B		NRM	0x1F		FOT	0x53
CGB	0x0C		CRG	0x20		OLM	0x54
CGBA	0x0D		IDR	0x21		SGM	0x55
CGU	0x0E		IRS	0x22		CMR	0x56
CGUA	0x0F		INR	0x23		CMC	0x57
GRS	0x10		INF	0x24		CMRJ	0x58
GRA	0x11		SUS	0x25		LOP	0x59
CCR	0x12		RES	0x26		APM	0x5A
COT	0x13		SAM	0x27		PRI	0x5B

Index values 0x2F-0x4E are user-defined for the ISUP message format table:

Message	Index		Message	Index		Message	Index
Message #1	0x2F		Message #14	0x3C		Message #27	0x49
Message #2	0x30		Message #15	0x3D		Message #28	0x4A
Message #3	0x31		Message #16	0x3E		Message #29	0x4B
Message #4	0x32		Message #17	0x3F		Message #30	0x4C
Message #5	0x33		Message #18	0x40		Message #31	0x4D
Message #6	0x34		Message #19	0x41		Message #32	0x4E
Message #7	0x35		Message #20	0x42			
Message #8	0x36		Message #21	0x43			
Message #9	0x37		Message #22	0x44			
Message #10	0x38		Message #23	0x45			
Message #11	0x39		Message #24	0x46			
Message #12	0x3A		Message #25	0x47			
Message #13	0x3B		Message #26	0x48			

ISUP Message ID This field specifies the ISUP Message ID for the specified ISUP message from the ISUP Message Format table. Valid entries for this field are listed by ISUP message name:

Message	ID		Message	ID		Message	ID
ACM	0x06		CRM	0xEA		OLM	0x30
ANM	0x09		CVR	0xEB		OPR	0xFE
APM	0x41		CVT	0xEC		PAM	0x28
BLA	0x15		DRS	0x27		PRI	0x42
BLO	0x13		EXM	0xED		REL	0x0C
CCL	0xFC		FAA	0x20		RES	0x0E
CFN	0x2F		FAC	0x33		RLC	0x10
CGB	0x18		FAR	0x1F		RSC	0x12
CGBA	0x1A		FOT	0x08		SAM	0x02
CGU	0x19		FRJ	0x21		SGM	0x38
CGUA	0x1B		GRA	0x29		SUS	0x0D
CON	0x07		GRS	0x17		UBA	0x16
CMC	0x49		IAM	0x01		UBL	0x14
CMRJ	0x4A		IDR	0x36		UCIC	0x2E
CMR	0x4B		INF	0x04		UPA	0x35
CPG	0x2C		INR	0x03		UPT	0x34
CQM	0x2A		IRS	0x37		USR	0x2D
CQR	0x2B		LOP	0x40			
CRA	0xE9		MPM	0xFD			
CRG	0x31		NRM	0x32			

Repeatable Optional Parameters/Optional Parameters Allowed

This field informs ISUP whether or not to look for or insert the optional parameters pointer octet in the ISUP message.

Some messages allow optional parameters, even though optional parameters are not defined (for example, the ITU Q.767 Release Complete (RLC) message).

- Optional parameters are required for the incoming message when you set the value of the *Optional Parameters Allowed* field to **0x01**. This is the default value. When you set this value to **0x01** and optional parameters are included in the message, the CSP automatically inserts the *End of Optional Parameters* octet. If the optional parameter pointer is missing in the incoming message, the CFN (Confusion message) will be returned.
- The maximum number of Optional Parameters is 52.
- For no optional parameters, set the value of the *Optional Parameters Allowed* field to **0x00** and the value of the *Number of Optional Parameters* field to **0x00**. You must complete this step to support the pre-1995 ANSI format for the Group Reset (GRS) and Group Reset Acknowledgement (GRA) messages.

Example: API message:

00 1f 00 6b 00 00 ff 00 01 08 01 00 10 17 00 00 01 16 01 01 01 00
00 02 4e 21 00 01 01 4e 22 00 01 01

00 1f 00 6b 00 00 ff 00 01 08 01 00 11 29 00 00 01 16 01 04 00 00
02 4e 21 00 01 01 4e 22 00 01 01

Example: SwitchKit API:

SS7ISUPMessageFormatConfig(note=0
StackID=0,
ISUPMessageIndex=10
ISUPmessageid=0x17
ISUPMessagePriority=0
NumMFPs=0,
Data=1:22:1:4:0:0:2:0x4e:0x21:00:01:01:0x4e:0x22:00:01:01);

SS7ISUPMessageFormatConfig(node=0,
StackID=0
ISUPMessageIndex=0x11
ISUPmessageid=0x29
ISUPMessagePriority=0
NumMFPs=0
Data=1:22:1:4:0:0:2:0x4e:0x21:00:00:01:01:0x4e:0x22:00:01:01);

- For Repeatable Optional Parameters, set the value of this field to the number of repeated optional parameters.

MSB	LSB
Bits 7-1 Number of Repeatable Optional Parameters Indicator	Bit 0 Optional Parameters Indicator

Options for Optional Parameters Allowed Indicator

- 0 Optional Parameters Not Allowed
- 1 Optional Parameters Allowed

Options for Number of Repeatable Optional Parameters

- 0 No Repeatable Optional Parameters
- 1 -127 Indicate the number Repeatable Optional Parameters

Maximum Length

A value of 0x00 for the maximum length fields indicates that no validation is performed for the maximum length of the parameter.

SS7 ISUP Message Query 0x0068

SwitchKit Name	SS7ISUPMessageQuery
Type	EXS API and SwitchKit API message
Description	SS7 ISUP Message Query 0x0068 This message reports the currently configured format of the specified ISUP message.
Sent by	Host
SwitchKit Code	C Structure

```
typedef struct {
    UBYTE StackID;
    UBYTE ISUPMessageIndex;
} XL_SS7ISUPMessageQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE StackID;
    UBYTE ISUPMessageIndex;
    UBYTE ISUPMessageID;
    UBYTE ISUPMessagePriority;
    UBYTE NumMFPs;
    UBYTE Data[217];
} XL_SS7ISUPMessageQueryAck;
```

C++ Class

```
class XL_C_SS7CICQuery : public XLC_OutboundMessage {
public:
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    int getDPC() const;
    void setDPC(int x);
    unsigned short getBaseCIC() const;
    void setBaseCIC(unsigned short x);
    UBYTE getNumCICs() const;
    void setNumCICs(UBYTE x);
    UBYTE getCallControlUserPart() const;
    void setCallControlUserPart(UBYTE x);
};
```

C++ Class Response

```
class XL_C_SS7CICQueryAck : public XLC_AcknowledgeMessage
{
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getStackID() const;
```

```

void setStackID(UBYTE x);
unsigned short getBaseCICNumber() const;
void setBaseCICNumber(unsigned short x);
unsigned short getBaseCICSpan() const;
void setBaseCICSpan(unsigned short x);
UBYTE getBaseCICChannel() const;
void setBaseCICChannel(UBYTE x);
UBYTE getNumCICs() const;
void setNumCICs(UBYTE x);
int getDPC() const;
void setDPC(int x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};
    
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x0068)	3, 4	Message Type (0x0068)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow	10	AIB (same as message)
	AE 0x08 SS7 Stack	:	ISUP Message Index
:	ISUP Message Index	:	ISUP Message ID
:	Checksum	:	ISUP Message Priority ANSI: T1.111.5 Appendix B ITU: This value has no meaning.
Response continued below.			
:	Number of Mandatory Fixed Parameters (MFP)		
:	MFP 1: ID		
:	MFP 1: Length		
:	MFP n: ID		
:	MFP n: Length		
:	Number of Mandatory Variable Parameters (MVP)		
:	MVP 1: ID		
:	MVP 1: Minimum Length		
:	MVP 1: Maximum Length (See Maximum Length at the end of this message description.)		
:	MVP n: ID		
:	MVP n: Minimum Length		

:	MVP n: Maximum Length (See Maximum Length at the end of this message description.)
:	<p>Optional Parameters Allowed</p> <p>This field informs ISUP whether or not to look for or insert the optional parameters pointer octet into the ISUP message.</p> <p>Some messages allow optional parameters, even though optional parameters are not defined (for example, the ITU Q.767 Release Complete (RLC) message).</p> <p>To include optional parameters, set the value of the Optional Parameters Allowed field to 0x01.</p> <p>For no optional parameters, set the value of the Optional Parameters Allowed field to 0x00 and the value of the Number of Optional Parameters field to 0x00.</p> <p>Refer to Repeatable Optional Parameters/Optional Parameters Allowed.</p>
:	<p>Number of Optional Parameters</p> <p>See Repeatable Optional Parameters/Optional Parameters Allowed at the end of this message.</p>
:	OP 1: ID
:	OP 1: Minimum Length
:	OP 1: Maximum Length (See Maximum Length at the end of this message description.)
:	OP n: ID
:	OP n: Minimum Length
:	OP n: Maximum Length (See Maximum Length at the end of this message description.)
:	Repeatable Optional Parameters
:	ROP 1: ID ROP n: ID
:	Number of TLVs
:	<p>TLV</p> <p>Both TLVs must be sent in the message and must be set to the same value.</p> <p>To remove an ISUP message, set both TLVs to 0x00.</p> <p>To add an ISUP message, set both TLVs to 0x01.</p> <p>0x4E24 Acceptance Rate</p> <p>0x4E22 Message Processing Supported</p>
:	Checksum

ISUP Message Index

This field specifies an index that is maintained for internal use by the CSP into the ISUP Message Format table for the specified ISUP message. The valid entries for this field are in numeric order.

Message	Index	Message	Index	Message	Index
IAM	0x00	LPA	0x14	CVT	0x28
ACM	0x01	CFN	0x15	CVR	0x29
ANM	0x02	UCIC	0x16	CRM	0x2A
CON	0x03	CQM	0x17	CRA	0x2B
REL	0x04	CQR	0x18	UPT	0x2C
RLC	0x05	FAR	0x19	UPA	0x2D
CPG	0x06	FAA	0x1A	EXM	0x2E
BLO	0x07	FRJ	0x1B	MPM	0x4F

Message	Index	Message	Index	Message	Index
BLA	0x08	USR	0x1C	OPR	0x50
UBL	0x09	FAC	0x1D	CCL	0x51
UBA	0x0A	PAM	0x1E	DRS	0x52
RSC	0x0B	NRM	0x1F	FOT	0x53
CGB	0x0C	CRG	0x20	OLM	0x54
CGBA	0x0D	IDR	0x21	SGM	0x55
CGU	0x0E	IRS	0x22	CMR	0x56
CGUA	0x0F	INR	0x23	CMC	0x57
GRS	0x10	INF	0x24	CMRJ	0x58
GRA	0x11	SUS	0x25	LOP	0x59
CCR	0x12	RES	0x26	APM	0x5A
COT	0x13	SAM	0x27	PRI	0x5B

Index values 0x2F-0x4E are user-defined for the ISUP message format table:

Message	Index	Message	Index	Message	Index
Message #1	0x2F	Message #14	0x3C	Message #27	0x49
Message #2	0x30	Message #15	0x3D	Message #28	0x4A
Message #3	0x31	Message #16	0x3E	Message #29	0x4B
Message #4	0x32	Message #17	0x3F	Message #30	0x4C
Message #5	0x33	Message #18	0x40	Message #31	0x4D
Message #6	0x34	Message #19	0x41	Message #32	0x4E
Message #7	0x35	Message #20	0x42		
Message #8	0x36	Message #21	0x43		
Message #9	0x37	Message #22	0x44		
Message #10	0x38	Message #23	0x45		
Message #11	0x39	Message #24	0x46		
Message #12	0x3A	Message #25	0x47		
Message #13	0x3B	Message #26	0x48		

ISUP Message ID

This field specifies the ISUP Message ID for the specified ISUP message from the ISUP Message Format table. Valid entries for this field are listed by ISUP message name:

Message	ID	Message	ID	Message	ID
ACM	0x06	CRM	0xEA	OLM	0x30
ANM	0x09	CVR	0xEB	OPR	0xFE
APM	0x41	CVT	0xEC	PAM	0x28
BLA	0x15	DRS	0x27	PRI	0x42
BLO	0x13	EXM	0xED	REL	0x0C
CCL	0xFC	FAA	0x20	RES	0x0E
CFN	0x2F	FAC	0x33	RLC	0x10
CGB	0x18	FAR	0x1F	RSC	0x12
CGBA	0x1A	FOT	0x08	SAM	0x02
CGU	0x19	FRJ	0x21	SGM	0x38
CGUA	0x1B	GRA	0x29	SUS	0x0D

Message	ID	Message	ID	Message	ID
CON	0x07	GRS	0x17	UBA	0x16
CMC	0x49	IAM	0x01	UBL	0x14
CMRJ	0x4A	IDR	0x36	UCIC	0x2E
CMR	0x4B	INF	0x04	UPA	0x35
CPG	0x2C	INR	0x03	UPT	0x34
CQM	0x2A	IRS	0x37	USR	0x2D
CQR	0x2B	LOP	0x40		
CRA	0xE9	MPM	0xFD		
CRG	0x31	NRM	0x32		

Repeatable Optional Parameters/Optional Parameters Allowed

This field informs ISUP whether or not to look for or insert the optional parameters pointer octet in the ISUP message.

Some messages allow optional parameters, even though optional parameters are not defined (for example, the ITU Q.767 Release Complete (RLC) message).

- To include optional parameters, set the value of the *Optional Parameters Allowed* field to **0x01**.
When you set this value to 0x01 and optional parameters are included in the message, the CSP automatically inserts the *End of Optional Parameters* octet.
- For no optional parameters, set the value of the *Optional Parameters Allowed* field to **0x00** and the value of the *Number of Optional Parameters* field to **0x00**.
- For Repeatable Optional Parameters, set the value of this field to the number of repeated optional parameters.

MSB	LSB
Bits 7-1 Number of Repeatable Optional Parameters Indicator	Bit 0 Optional Parameters Indicator

Options for Optional Parameters Allowed Indicator

- 0 Optional Parameters Not Allowed
- 1 Optional Parameters Allowed

Options for Number of Repeatable Optional Parameters

- 0 No Repeatable Optional Parameters
- 1 -127 Indicate the number Repeatable Optional Parameters

Maximum Length

A value of 0x00 for the maximum length fields indicates that no validation is performed for the maximum length of the parameter.

SS7 SCCP/TCAP Configure 0x0077

SwitchKit Name SS7SCCPTCAPConfig

Type EXS API and SwitchKit API message

Description **SS7 SCCP/TCAP Configure 0x0077**

This message is sent to configure options for SS7 SCCP/TCAP.

For SwitchKit

Those options include:

- Local SSN
- Adjacent Translator
- Other Concerned Point Code
- SCCP Default Parameter Table
- SSN Default Parameter Table
- Network DPC/SSN
- SCCP/TCAP host configuration

In a SwitchKit environment, this message must be sent by an application or by SwitchManager with ConfigType = 1, Data = [2:1] specifying “SCCP route message to TCAP”. Without sending this message first and receiving a positive acknowledgment it is impossible to add or specify a node as a TCAP node.

Sent by Host (SwitchManager for SwitchKit)

Overview of message The following table provides an overview of this message. The table following it provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0077)	3, 4	Message Type (0x0077)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8, 9	Status MSB, LSB
:	Address Method		
	Number of AEs to follow	10	Checksum
	AEs		
9	Config Type		
10	Number of TLVs		
11	TLVs		
:	Data[0]		
:	:		
:	Checksum		

Configuration `SS7SCCPTCAPConfig (`
 Node = integer,
 StackID = integer,
 ConfigType = integer,
 Data = byte array);

C Structure `typedef struct {`
 UBYTE StackID;
 UBYTE ConfigType;
 UBYTE Data[222];
 } **XL_SS7SCCPTCAPConfig;**

C++ Class `class XLC_SS7SCCPTCAPConfig : public XLC_OutboundMessage`
 {
 public:
 UBYTE getStackID() const;
 void setStackID(UBYTE x);
 UBYTE getConfigType() const;
 void setConfigType(UBYTE x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
 };

**EXS API Hex Format -
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0077)	3, 4	Message Type (0x0077)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID

<p>8 : AIB Address Method 0x00 - Individual AEs</p>		<p>8, 9</p>	<p>Status MSB - 0x55 LSB - As follows: 01 - Invalid CFG Type 02 - Subsystem Configured 03 - Subsystem Not Configured 04 - Subsystem Allowed 05 - Subsystem Not Allowed 06 - Invalid Option 07 - Exceed Max SSN Host 08 - Invalid Parameter 09 - Invalid Parameter Length 0A - Parameter Configured 0B - Parameter Not Configured 0C - Invalid SCCP Upper Layer 0D - Exceed Max Active Subsystem 0E - DPC Not Configured 0F - Exceed Max Replicates 10 - Invalid Number of Replicates 11 - Replicate Not Configured 12 - Exceed Max Active SSN 13 - Exceed Max SSN PPC 14 - ESSG Subsystem Not Configured 15 - ESSG SSN PPC Exist 16 - OPR Not Perm for ESSG SSN DPC SSN 17 - Local Host Not Allowed 19 - Invalid Tag Type 1A - SCCP not configured for this stack 1B - GTAI length is out of the range configured for the group 1D - GT Group hasn't been configured 1E - GT Group ID is already occupied 1F - GT Group with the same stack ID and selector already exists 20 - Group ID exceed maximum group ID 21 - The group parameters are not matched with parameters stored in the CSP. 22 - The minimum/maximum digits number in GT group is illegal. 23 - GT Entry already in Pending DEL state 24 - GT already exists in the group. 25 - GT does not exist. 26 - The memory storing translation result with global title has been used up. 27 - GT Entries have been used up. 28 - GT Entry is found in standby index table. 29 - GT Entry is found in active index table. 2A - GT Group is configured but not for this stack. 2B - Configuration is unreasonable. 1. Translation result is route on GT but DPC is equal to OPC. 2C - CPU load is not suitable to configuration. 2D - Index table needs to be built.</p>
	<p>Number of AEs to follow</p>	<p>10</p>	<p>Checksum</p>
<p>AE 0x08 SS7 Stack</p>			

SS7 SCCP/TCAP Configure 0x0077

:	<p>Config Type</p> <ul style="list-style-type: none"> 0x01 Local SSN 0x02 Adjacent Translator 0x03 Other Concerned Point Code 0x04 SCCP Default Parameter Table 0x05 SSN Default Parameter Table 0x07 Network DPC/SSN 0x08 SCCP/TCAP host configuration 0x09 SCCP Replicated SSN 0x0C TLV Format Configuration Contents
:	<p>Number of TLVs</p>
:	<p>TLVs</p> <p>The Play File Start (0x011B) supports the following new TLV.</p> <ul style="list-style-type: none"> 0x09CB Delete Global Title Group 0x09CC Add Global Title Entry 0x09CD Delete Global Title Entry 0x09CE Build Index Table <p>Refer to <i>Chapter 4, Tag Length Blocks</i> in the <i>API Reference</i>.</p> <p>See also, <i>Global Title Translation (GTT) Configuration Samples</i> in the <i>Developer's Guide: Common Channel Signaling</i>.</p>

:	<p>Data[0] The Data is dependent on the Config Type field (0x01 - 0x09)</p> <p>0x01 Local SSN</p> <p>Data[0] Subsystem Number</p> <p>Data[1] Option</p> <p style="padding-left: 40px;">0x01 Add SSN</p> <p style="padding-left: 40px;">0x02 Delete SSN</p> <p style="padding-left: 40px;">0x03 Modify SSN</p> <p>Data[2] Routing Flag</p> <p style="padding-left: 40px;">Bit 0 - Specify the direct upper layer of the SCCP (TCAP or host application.)</p> <p style="padding-left: 80px;">0 - SCCP Route network message directly to host</p> <p style="padding-left: 80px;">1 - SCCP Route network message to TCAP</p> <p style="padding-left: 40px;">Bits 1 and 3-7 - Reserved for SASSI and ESSG use. Not used in current release. Default is 0.</p> <p style="padding-left: 40px;">Bit 2 - Specify whether local GTT function is allowed to use when the message is originated from this SSN. (See note below.)</p> <p style="padding-left: 80px;">0 - Disable GTT when message is originated from the local SSN.</p> <p style="padding-left: 80px;">1 - Enable GTT when message is originated from the local SSN.</p> <p>Data[3] Reserved</p> <p>Note: Bit 2 only enables/disables GTT when the message is originated from the local SSN. When a message is received from the network and it needs GTT, the GTT function is always triggered. This bit provides backward compatibility to those host applications that do not support GTT.</p>
---	---

0x02 Adjacent Translator

Data[0] Subsystem Number

Data[1] Option

0x01 Add Adjacent Translator

0x02 Delete Adjacent Translator

Data[2-5] Adjacent Translator

0x03 Other Concerned Point Code

Data[0] Subsystem Number

Data[1] Option

0x01 Add Other Concerned Point Code

0x02 Delete Other Concerned Point Code

Data[2-5] Other Concerned Point Code

0x04 SCCP/TCAP Default Parameter Table

Data[0] Option

0x01 Add/Modify SCCP/TCAP Default Parameter

0x02 Delete SCCP/TCAP Default Parameter

Data[1-2] Parameter Type

Data[3-4] Parameter Length

Data[5+] Parameter Value

0x05 SSN Default Parameter Table

Data[0] Subsystem Number

Data[1] Option

0x01 Add/Modify SSN Default Parameter

0x02 Delete SSN Default Parameter

Data[2-3] Parameter Type

0x0066 Default Calling Party Address

0x006A Default Quality of Service

0x0074 Default MTP dpc (used when DPC is not in CDPA)

Data[4-5] Parameter Length

Data[6+] Parameter Value

0x07 Network DPC/SSN

Data[0] Local Subsystem Number

Data[1] Option
0x01 Add Network DPC/SSN
0x02 Delete Network DPC/SSN

Data[2-5] DPC

Data[6] Remote SSN Number

Use this configuration if a certain subsystem needs to talk to a network DPC/SSN directly (route on DPC/SSN). You do not need to configure this if route on global title. A network DPC/SSN could be a remote or local subsystem.

0x08 SCCP/TCAP host configuration

Data[0] Subsystem Number

Data[1] Option
0x01 Add a host
0x02 Delete a host

Data[2] Local/Matrix host
0xFE Local host
0xFF Matrix host

Data[3] Host Port ID (0-5)

Port IDs 0-5 correspond to Ports 0x3142-0x3147.

NOTE: Only one host can be configured for each stack/subsystem. The default host is the matrix primary host. Also, an SS7 card supports only one port: 0x3142. Note however that it is possible to have an additional direct connection between the second host and the TCAP on the SS7 card using the CCS I/O card.

	<p>0x09 SS7 SCAP CFG Replicated SSN</p> <p>Data[0] Subsystem Number</p> <p>Data[1] Option</p> <p style="padding-left: 40px;">0x01 Add Replicated SSN 0x02 Delete Replicated SSN</p> <p>Data[2] Number of Replicates</p> <p>Data[3-6] DPC</p> <p>Data[7] Subsystem Number</p> <p>Data[8...] Next DPC and Subsystem Number</p> <p>0x0C TLV Format Configuration Contents</p> <p style="padding-left: 40px;">0x09CB Delete Global Title Group 0x09CC Add Global Title Entry 0x09CD Delete Global Title Entry 0x09CE Build Index Table</p>
:	Checksum

SS7 SCCP/TCAP Query 0x0078

SwitchKit Name	SS7SCCPTCAPQuery
Type	EXS API and SwitchKit API message
Description	<p>SS7 SCCP/TCAP Query 0x0078</p> <p>This message is sent to configure query information about SS7 SCCP/TCAP configuration.</p>
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE StackID;
    UBYTE reserved18[29];
    UBYTE QueryType;
    UBYTE Data[222];
} XL_SS7SCCPTCAPQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE StackID;
    UBYTE QueryType;
    UBYTE Data[220];
} XL_SS7SCCPTCAPQueryAck;
```

C++ Class

```
class XL_SS7SCCPTCAPQuery : public XLC_OutboundMessage {
public:
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    UBYTE getQueryType() const;
    void setQueryType(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

C++ Class Response

```
class XL_SS7SCCPTCAPQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    UBYTE getQueryType() const;
    void setQueryType(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
};
```

```
void setData(UBYTE *x);
};
```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x0078)	3, 4	Message Type (0x0078)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8, 9	Status MSB, LSB
:	Address Method		
	Number of AEs to follow	10	AIB (same as message)
	AEs	:	Query Type
:	Query Type	:	Data[0]
:	Number of TLVs	:	Checksum
:	TLVs		
:	Data[0]		
:	:		
:	Checksum		

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1	Length, MSB	1	Length, MSB
2	Length, LSB (N)	2	Length, LSB (N)
3	Message Type, MSB (0x00)	3	Message Type, MSB (0x00)
4	Message Type, LSB (0x78)	4	Message Type, LSB (0x78)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID

<p>8 : AIB Address Method 0x00 - (Individual AEs)</p>		<p>8, 9</p>	<p>Status MSB - 0x55 LSB - As follows: 01 - Invalid CFG Type 02 - Subsystem Configured 03 - Subsystem Not Configured 04 - Subsystem Allowed 05 - Subsystem Not Allowed 06 - Invalid Option 07 - Exceed Max SSN Host 08 - Invalid Parameter 09 - Invalid Parameter Length 0A - Parameter Configured 0B - Parameter Not Configured 0C - Invalid SCCP Upper Layer 0D - Exceed Max Active Subsystem 0E - DPC Not Configured 0F - Exceed Max Replicates 10 - Invalid Number of Replicates 11 - Replicate Not Configured 12 - Exceed Max Active SSN 13 - Exceed Max SSN PPC 14 - ESSG Subsystem Not Configured 15 - ESSG SSN PPC Exist 16 - OPR Not Perm for ESSG SSN DPC SSN 17 - Local Host Not Allowed 19 - Invalid Tag Type 1A - SCCP not configured for this stack 1B - GTAI length is out of the range configured for the group 1D - GT Group hasn't been configured 1E - GT Group ID is already occupied 1F - GT Group with the same stack ID and selector already exists 20 - Group ID exceed maximum group ID 21 - The group parameters is not matched with parameters stored in the switch. 22 - The minimum/maximum digits number in GT group is illegal. 23 - GT entry already in Pending DEL state 24 - GT already exists in the group. 25 - GT does not exist. 26 - The memory storing translation result with global title has been used up. 27 - GT entries have been used up. 28 - GT entry is found in standby index table 29 - GT Entry is found in active index table 2A - GT Group is configured but not for this stack. 2B - Configuration is unreasonable. 1. Translation result is "route on GT but DPC is equal to OPC." 2C - CPU load is not suitable to config. 2D - Index table needs to be built.</p>
	<p>Number of AEs to follow</p>	<p>10</p>	<p>AIB (same as message)</p>
<p>AE 0x08 SS7 Stack</p>		<p>:</p>	<p>Query Type</p>

SS7 SCCP/TCAP Query 0x0078

	<p>Query Type</p> <ul style="list-style-type: none"> 0x01 Local SSN Table 0x02 SCCP/TCAP Default Parameter 0x03 SSN Default Parameter 0x05 TCAP Active Transactions 0x06 TCAP Active Operations 0x08 SCCP Replicated SSN 0x0A Associated subsystems for a stack 0x0B Number of active dialog and invocations 0x0C TLV Format Configuration Contents 	:	Number of TLVs
Number of TLVs		:	<p>TLVs</p> <ul style="list-style-type: none"> 0x09D1 Global Title Group Query Response 0x09D2 Global Title Entry Query Response

	<p>TLVs 0x09CF Global Title Group Query 0x09D0 Global Title Entry Query</p> <p>Refer to <i>Chapter 4, Tag Length Blocks</i> in the <i>API Reference</i>. See also, <i>Global Title Translation (GTT) Configuration Samples</i> in the <i>Developer's Guide: Common Channel Signaling</i>.</p>	<p>:</p>	Data[0] (See info below this table)
		<p>:</p>	Checksum
<p>:</p>	<p>Data[0] The Data is dependent on the <i>Query Type</i>.</p> <p>0x01 Local SSN Table Data[0] Subsystem Number Data[1] Subsystem Status</p> <p>0x00 Prohibited 0x01 Allowed</p> <p>0x02 SCCP/TCAP Default Parameter Data[0-1] Parameter Type</p> <p>0x03 SSN Default Parameter Data[0] Subsystem Number Data[1-2] Parameter Type Refer to the <i>Developer's Guide: Common Channel Signaling</i> for Parameter Information.</p> <p>0x05 TCAP Active Transactions Data[0] Subsystem Number Data[1-4] Dialog ID</p> <p>0x06 TCAP Active Operations Data[0] Subsystem Number Data[1-4] Dialog ID Data[5, 6] Invoke ID</p> <p>0x0B Number of active dialogues and invocations No data.</p> <p>0x0C TLV Format Configuration Contents</p>	<p>:</p>	Checksum

Response Data

The Response Data is dependent on the *Query Type*.

0x01 Local SSN Table

Data[0]	Subsystem Number
Data[1]	Subsystem Status
	0x00 Prohibited
	0x01 Allowed
	Number of Network DPC/SSNs (1 byte)
	DPC 1 (4 bytes)
	Network SSN 1 (1 byte)
	Network SSN 1 Status (1 byte)
	:
	DPC <i>n</i> (4 bytes)
	Network SSN <i>n</i>
	Network SSN <i>n</i> Status (1 byte)
	Number of Hosts (1 byte)
	Local/Matrix Host (1 byte)
	Port ID (1 byte)
Data[2]	Routing Flag
	0x00 SCCP Route network message directly to host
	0x01 SCCP Route network message to TCAP
Data[3]	Number of Adjacent Translators
Data[4-7]	Adjacent Translator 1 (4 bytes)
	:
	Adjacent Translator <i>n</i> (4 bytes)
	Number of Other Concerned Point Codes (1 byte)
	Other Concerned Point Code 1 (4 bytes)
	:
	Other Concerned Point Code <i>n</i> (4 bytes)
	Number of Network DPC/SSNs (1 byte)
	DPC 1 (4 bytes)
	Network SSN 1 (1 byte)
	Network SSN 1 Status (1 byte)
	0x00 Prohibited
	0x01 Allowed
	:
	DPC <i>n</i> (4 bytes)
	Network SSN <i>n</i>
	Network SSN <i>n</i> Status (1 byte)
	0x00 Prohibited
	0x01 Allowed
	Number of Hosts (1 byte)
	Local/Matrix Host (1 byte)
	Port ID (1 byte)

0x02 SCCP/TCAP Default Parameter Table

Data[0-1] Parameter Type
Data[2-3] Parameter Length
Data[4+] Parameter Value

Refer to the *Developer's Guide: Common Channel Signaling* for Parameter Information.

0x03 SSN Default Parameter Table

Data[0] Subsystem Number
Data[1-2] Parameter Type
Data[3-4] Parameter Length
Data[5+] Parameter Value

Refer to the *Developer's Guide: Common Channel Signaling* for Parameter Information.

0x05 TCAP Active Transactions

Data[0] Subsystem Number
Data[1-4] Dialog ID (4 bytes)
Data[5-8] Transaction Originating Transaction ID (4 bytes)
Data[9-12] Transaction Responding Transaction ID (4 bytes)
Data[13] Transaction State

ANSI

The following *Transaction State* values apply for ANSI:

- 0x00 Idle
- 0x01 Query Without Permission package has been sent
- 0x02 Query With Permission package has been sent
- 0x03 Query Without Permission package has been received
- 0x04 Query With Permission package has been received
- 0x05 Conversation With Permission package has been received
- 0x06 Conversation Without Permission package has been received
- 0x07 Conversation Without Permission package has been sent
- 0x08 Conversation With Permission package has been sent

ITU

The following *Transaction State* values apply for ITU:

- 0x00 Idle
- 0x01 Initiation Received
- 0x02 Initiation Sent
- 0x03 Active

The following bytes are the stored CGPA for this transaction:

Data[14-15] CGPA Parameter Type
Data[16-17] Parameter Length
Data[18+] Parameter Value

The following bytes are the stored CGPA for this transaction:

Data[:] CDPA Parameter Type
Data[:] Parameter Length
Data[:+] Parameter Value
Data[:]:

Refer to the *Developer's Guide: Common Channel Signaling* for Parameter Information.

Data[:]Reserved

0x06 TCAP Active Operations

Data[0] Subsystem Number

Data[1-5]Dialog ID (4 bytes)

Data[6-7]Operation Invoke ID

Data[8-9]Operation Correlation ID

The MSB is always 0x00 unless the Correlation ID is not available, in which case it is 0xFF)

Data[10] Operation State

ANSI

The following *Operation State* values apply for ANSI:

0x00 Idle The operation is not initiated

0x01 Operation is Pending The state machine is waiting for an incoming component from the network

0x02 Operation is in Progress The state machine is expecting the host to send a responding component

0x03 Operation Local Pending All components required for this operation are stored in TCAP.

A Dialog primitive must be sent to send the components to the network.

In this state, any component received for this operation from the network or the host will be rejected.

ITU

The following *Operation State* values apply for ITU:

0x00 Idle

0x01 Operation Sent Class 1

0x02 Wait for Reject

0x03 Operation Sent Class 2

0x04 Operation Sent Class 3

0x05 Operation Sent Class 4

Data[9-12]Reserved

0x0B Number of active dialogs and invocations

Data[0-3] number of active dialogs

Data[4-7] number of active invocations

Data[8-11] next available dialog ID stack will use for the next incoming new dialog

0x0C TLV Format Configuration Contents

0x09D1 Global Title Group Query Response

0x09D2 Global Title Entry Query Response

SS7 Signaling Link Configure 0x005E

SwitchKit Name SS7SignalingLinkConfig

Type EXS API and SwitchKit API message

Description **SS7 Signaling Link Configure 0x005E**

This message defines a signaling link and assigns it to a link set. You can also use this message to subsequently deconfigure a signaling link.

Sent by Host

Example Message (Socket Log Output for SwitchKit)

The following example message defines signaling link 0x05 on a primary SS7 card. The signaling link is assigned to link set 0x23. The link's data rate is 64 Kbps rate and the electrical interface is V.35 DTE (Data Terminal Equipment).

```
00 14 00 5E 00 00 FF 00 02 1F 03 00 23 05 0D 03 00 01 17 01 00 01
```

SwitchKit Code **Configuration**

```
SS7SignalingLinkConfig (
    Node = integer,
    StackID = integer,
    LinkSetID = integer,
    LinkID = integer,
    Span = integer,
    Channel = integer,
    SLC = integer,
    DataRate = integer,
    ElectricalInterface = integer,
```

C Structure

```
typedef struct {
    UBYTE AddrInfo[4];
    UBYTE StackID;
    UBYTE LinkSetID;
    UBYTE LinkID;
    unsigned short Span;
    UBYTE Channel;
    UBYTE SLC;
    UBYTE DataRate;
    UBYTE ElectricalInterface;
    UBYTE Reserved;
} XL_SS7SignalingLinkConfig;
```

C++ Class

```
class XLC_SS7SignalingLinkConfig : public
    XLC_OutboundMessage {
```

```

public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    UBYTE getLinkSetID() const;
    void setLinkSetID(UBYTE x);
    UBYTE getLinkID() const;
    void setLinkID(UBYTE x);
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getChannel() const;
    void setChannel(UBYTE x);
    UBYTE getSLC() const;
    void setSLC(UBYTE x);
    UBYTE getDataRate() const;
    void setDataRate(UBYTE x);
    UBYTE getElectricalInterface() const;
    void setElectricalInterface(UBYTE x);
    UBYTE getReserved() const;
    void setReserved(UBYTE x);
};
    
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x005E)	3, 4	Message Type (0x005E)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB)
:	Address Method 0x00 - Individual AEs	10	Checksum
	Number of AEs to follow 0x02		
	AEs See Address Element following this table.		
:	Signaling Link Code (SLC) The SLC must be the same on both ends of the link (0x00–0x0F).		

SS7 Signaling Link Configure 0x005E

:	<p>Data Rate</p> <p>0x00 64 Kbps 0x01 56 Kbps for non-V.35 applications 0x80 V.35, 56 Kbps 0x81 48 Kbps for non-V.35 applications 0xC0 V.35, 48 Kbps</p>
:	<p>Electrical Interface</p> <p>0xn0 No timing (Not a Multi-Protocol I/O link) 0xn1 V.35, DTE (Data Terminal Equipment) 0xn2 V.35, DCE (Data Communications Equipment)</p> <p>The value for n is as follows:</p> <p>Bit-4 (0x1n) is used to invert the data stream on non-MPIO (normal DS0) configurations. Bit 7 (0x8n) is used to invert the data stream on MPIO configurations. The options are as follows:</p> <p>Invert data stream on non-MPIO configurations: 0x11 - V.35, DTE (Data Terminal Equipment) 0x12 - V.35, DCE (Data Communications Equipment)</p> <p>Invert data stream on MPIO configurations: 0x81 - V.35, DTE (Data Terminal Equipment) 0x82 - V.35, DCE (Data Communications Equipment)</p>
:	Checksum

Address Element To configure and deconfigure a link, use a combination of the Link ID and Channel address type, as follows:

Byte	AIB Field Description
2	Address Element (0x1F)
3	Data Length (0x03)
4	Data[0] Stack ID
5	<p>Data[1] Link Set ID</p> <p>This field indicates the signaling link set to which this link is being assigned.</p> <p>0x00-0x23 Signaling Link Sets 0–35 0xFF Deconfigure the specified signaling link</p>
6	<p>Data[2] Link ID</p> <p>This field indicates values that are assigned by the host to links on an SS7 card. Values for this field depend on how many links are supported by the card and whether it is a primary or secondary card. With a 16-link card, valid entries for this field are as follows:</p> <p>0x00-0x0F Primary SS7 Card: 0x10-0x1F Secondary SS7 Card:</p>
7	Address Element (0x0D)
8	Data Length (0x03)

SS7 Signaling Link Configure 0x005E

Byte	AIB Field Description
9	Data[0] – Logical Span ID, MSB
10	Data[1] – Logical Span ID, LSB
11	Data[2] – Channel

SS7 Signaling Link Query 0x0065

SwitchKit Name	SS7SignalingLinkQuery
Type	EXS API and SwitchKit API message
Description	SS7 Signaling Link Query 0x0065 This message queries information on a signaling link.
Sent by	Host
SwitchKit Code	C Structure

```
typedef struct {
    UBYTE StackID;
    UBYTE LinkSetID;
} XL_SS7SignalingLinkQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE StackID;
    UBYTE LinkSetID;
    UBYTE NumLinks;
    UBYTE Data[220];
} XL_SS7SignalingLinkQueryAck;
```

C++ Class

```
class XLC_SS7SignalingLinkQuery : public
    XLC_OutboundMessage {
public:
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    UBYTE getLinkSetID() const;
    void setLinkSetID(UBYTE x);
};
```

C++ Class Response

```
class XLC_SS7SignalingLinkQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    UBYTE getLinkSetID() const;
    void setLinkSetID(UBYTE x);
    UBYTE getNumLinks() const;
    void setNumLinks(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
```

};

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)																			
Byte	Field Description	Byte	Field Description																		
0	Frame (0xFE)	0	Frame (0xFE)																		
1	Length (0x0007)	1	Length (0x00NN)																		
3	Message Type (0x0065)	3	Message Type (0x0065)																		
5	Reserved (0x00)	5	Reserved (0x00)																		
6	Sequence Number	6	Same Sequence Number																		
7	Logical Node ID	7	Logical Node ID																		
8	AIB	8, 9	Status MSB, LSB																		
:	Address Method 0x00 - Individual AEs																				
	Number of AEs to follow	10	AIB (same as message)																		
	AEs 0x1E SS7 Link Set 0x0D Channel																				
:	Checksum																				
Response continued below.																					
:	Number of Links																				
:	Link n: ID																				
:	Link n: Status This field is a bit mask. The meaning of each bit is as follows:																				
	<table border="0"> <tr> <td><u>Bit</u></td> <td><u>Status</u></td> </tr> <tr> <td>0</td> <td>Available</td> </tr> <tr> <td>1</td> <td>Failed</td> </tr> <tr> <td>2</td> <td>Inhibit Locally</td> </tr> <tr> <td>3</td> <td>Inhibit Remotely</td> </tr> <tr> <td>4</td> <td>Remote Blocked</td> </tr> <tr> <td>5-7</td> <td>Reserved</td> </tr> </table>	<u>Bit</u>	<u>Status</u>	0	Available	1	Failed	2	Inhibit Locally	3	Inhibit Remotely	4	Remote Blocked	5-7	Reserved						
<u>Bit</u>	<u>Status</u>																				
0	Available																				
1	Failed																				
2	Inhibit Locally																				
3	Inhibit Remotely																				
4	Remote Blocked																				
5-7	Reserved																				
:	Signaling Link Code (n) The signaling link code (SLC) must be the same on both ends of the link (0x00–0x0F).																				
:	Data Rate (n) This field specifies a bit mask that determines the data rate. If any one bit is set, the data rate will be 56 Kbps. If any two bits are set, the data rate will be 48 Kbps. The most common settings for this field are:																				
	<table border="0"> <tr> <td>0x00</td> <td>64 Kbps</td> <td></td> </tr> <tr> <td>0x01</td> <td>56 Kbps</td> <td>for non-V.35 applications (mask off least significant bit)</td> </tr> <tr> <td>0x80</td> <td>V.35, 56 Kbps</td> <td>(mask off most significant bit)</td> </tr> <tr> <td>0x81</td> <td>48 Kbps</td> <td>for non-V.35 applications</td> </tr> <tr> <td></td> <td></td> <td>(mask off least significant and most significant bits)</td> </tr> <tr> <td>0xC0</td> <td>V.35, 48 Kbps</td> <td>(mask off 2 most significant bits)</td> </tr> </table>	0x00	64 Kbps		0x01	56 Kbps	for non-V.35 applications (mask off least significant bit)	0x80	V.35, 56 Kbps	(mask off most significant bit)	0x81	48 Kbps	for non-V.35 applications			(mask off least significant and most significant bits)	0xC0	V.35, 48 Kbps	(mask off 2 most significant bits)		
0x00	64 Kbps																				
0x01	56 Kbps	for non-V.35 applications (mask off least significant bit)																			
0x80	V.35, 56 Kbps	(mask off most significant bit)																			
0x81	48 Kbps	for non-V.35 applications																			
		(mask off least significant and most significant bits)																			
0xC0	V.35, 48 Kbps	(mask off 2 most significant bits)																			

SS7 Signaling Link Query 0x0065

:	<p>Electrical Interface (n)</p> <p>0x00 No timing (Not a V.35 link) 0x01 V.35, DTE (Data Terminal Equipment) 0x02 V.35, DCE (Data Communications Equipment)</p> <p>In addition, Bit 8 specifies data inversion of the V.35 data stream. The data inversion is performed on the SS7 MP I/O card.</p> <p>0x81 Invert Transmit and Receive Data</p> <p>For example, if the Electrical Interface Byte is 0x81, data inversion is performed on both Transmit and Receive Data, and the interface is V.35 DTE.</p>
:	<p>AIB (Individual AEs) Span/Channel that Link (n) is on. 0x0D Channel</p>
:	Checksum

SS7 Signaling Link Set Configure 0x005D

SwitchKit Name SS7SignalingLinkSetConfig

Type EXS API and SwitchKit API message

Description **SS7 Signaling Link Set Configure 0x005D**

This message allows the host to define and subsequently deconfigure a signaling link set, which is a load-sharing collection of signaling links connected to an adjacent signaling node. The link set and stack must be configured before the signaling links.

You can also set this message to deconfigure a link set.

Example Message (Socket Log Output for SwitchKit)

The following example message defines a signaling link set with an OPC for a signaling stack that uses 24-bit point codes (ANSI).

```
00 0F 00 5D 00 00 FF 00 01 1E 02 00 00 00 20 00 00
```

Sent by Host

SwitchKit Code **Configuration**

```
SS7SignalingLinkSetConfig (
    Node = integer,
    StackID = integer,
    LinkSetID = integer,
    APC = integer);
```

C Structure

```
typedef struct {
    UBYTE StackID;
    UBYTE LinkSetID;
    int APC;
} XL_SS7SignalingLinkSetConfig;
```

C++ Class

```
class XLC_SS7SignalingLinkSetConfig : public
    XLC_OutboundMessage {
public:
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    UBYTE getLinkSetID() const;
    void setLinkSetID(UBYTE x);
    int getAPC() const;
    void setAPC(int x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type, MSB (0x005D)	3, 4	Message Type, MSB (0x005D)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB)
:	Address Method 0x00 - Individual AEs	10	Checksum
	Number of AEs to follow		
	AEs 0x1E SS7 Link Set		
:	<p>APC (four bytes) SS7 network address of the adjacent signaling node. Every point code in the API messages will always be 4 bytes long with the first byte (MSB) as zero. See the SS7 section of the <i>API Developer's Guide: CCS</i> for an explanation of how to convert the 4 bytes into ITU and ANSI point codes.</p> <p>0x00000000–0x00FFFFFF The APC must be in this range for signaling stacks using a 24-bit point code (for example ANSI).</p> <p>0x00000000–0x00003FFF The APC must be in this range for signaling stacks using a 14-bit point code (for example, ITU).</p> <p>0xFFFFFFFF Indicates deconfiguration of the specified link set. You can perform deconfiguration only after each link and route previously assigned to the link set has been deconfigured.</p>		
:	Checksum		

SS7 Signaling Link Set Query 0x0064

SwitchKit Name' SS7SignalingLinkSetQuery

Type EXS API and SwitchKit API message

Description **SS7 Signaling Link Set Query 0x0064**

This message allows the host to query information on a signaling link set.

NOTE: Every point code in API messages will always be 4 bytes long with the first byte MSB as zero. See *Signaling System 7* in the *API Developer's Guide: CCS* for an explanation of how to convert the 4 bytes for APC into ITU and ANSI point codes.

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE StackID;
} XL_SS7SignalingLinkSetQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE StackID;
    UBYTE NumLinkSets;
    UBYTE Data[220];
} XL_SS7SignalingLinkSetQueryAck;
```

C++ Class

```
class XLC_SS7SignalingLinkSetQuery : public
    XLC_OutboundMessage {
public:
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
};
```

C++ Class Response

```
class XLC_SS7SignalingLinkSetQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    UBYTE getNumLinkSets() const;
    void setNumLinkSets(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
```

};

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x0007)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x0064)	3, 4	Message Type (0x0064)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB)
:	Address Method 0x00 - Individual AEs	10	AIB (same as message)
	Number of AEs to follow		
	AE 0x08 SS7 Stack		
:	Checksum		
	Response continued below.		
:	Number of Link Sets (n)		
:	Link Set n: ID		
:	Link Set n: Status 0x00 Unavailable 0x01 Available		
:	Link Set n: APC APC: (four bytes) Every point code in the API messages will always be 4 bytes long with the first byte (MSB) as zero. See the SS7 section of the <i>Developer's Guide: Common Channel Signaling</i> for an explanation of how to convert the 4 bytes into ITU and ANSI point codes.		
:	Checksum		

SS7 Signaling Route Configure 0x005F

SwitchKit Name	SS7SignalingRouteConfig
Type	EXS API and SwitchKit API message
Description	<p>SS7 Signaling Route Configure 0x005F</p> <p>This message allows the host to assign or to deconfigure a signaling route to reach a specific DPC. Each assigned route to a DPC is given an associated priority route selection in the presence of multiple available routes for a DPC.</p>
Sent by	Host

SwitchKit Code **Configuration**

```
SS7SignalingRouteConfig (
    Node = integer,
    StackID = integer,
    Destination = integer,
    RouteID = integer,
    DPC = integer,
    LinkSetID = integer,
    Priority = integer,
    CombinedLinkSetRef = integer);
```

C Structure

```
typedef struct {
    UBYTE StackID;
    unsigned short Destination;
    unsigned short RouteID;
    int DPC;
    UBYTE LinkSetID;
    UBYTE Priority;
    unsigned short CombinedLinkSetRef;
} XL_SS7SignalingRouteConfig;
```

C++ Class

```
class XLC_SS7SignalingRouteConfig : public
    XLC_OutboundMessage {
public:
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    unsigned short getDestination() const;
    void setDestination(unsigned short x);
    unsigned short getRouteID() const;
    void setRouteID(unsigned short x);
    int getDPC() const;
    void setDPC(int x);
    UBYTE getLinkSetID() const;
    void setLinkSetID(UBYTE x);
    UBYTE getPriority() const;
    void setPriority(UBYTE x);
```

```
unsigned short getCombinedLinkSetRef() const;
void setCombinedLinkSetRef(unsigned short x);
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x005F)	3, 4	Message Type (0x005F)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB Address Method 0x00 - Individual AEs	8	Status (MSB, LSB) The meaning of these fields depends upon the value of the <i>Entity</i> field in the message. If the MSB of the Status field is 0x01 (NACK), the LSB indicates the NACK reason. 0x01 SS7 invalid combined linkset reference. The Route ID given as the reference is not configured.
		9	Checksum
Number of AEs to follow AEs 0x20 SS7 Destination/Route Note: After you issue the first <i>SS7 Signaling Route Configure</i> message for a DPC, all subsequent <i>SS7 Signaling Route Configure</i> messages must use the same Destination ID within the AE for additional routes to the same DPC. (There is a unique association between DPC and an assigned Destination ID.)			
:	DPC (four bytes) Every point code in the API messages will always be 4 bytes long with the first byte (MSB) as zero. See the SS7 section of the <i>API Developer's Guide: CCS</i> for an explanation of how to convert the 4 bytes into ITU and ANSI point codes. 0x00000000–0x00FFFFFF The DPC must be in this range for signaling stacks using 24-bit point codes (for example, ANSI). 0x00000000–0x00003FFF The DPC must be in this range for signaling stacks using 14-bit point codes (for example, ITU). 0xFFFFFFFF Indicates de-assignment of the specified destination and all routes to the specified destination. This is the only method that can be used to de-assign a destination. If using the BT IUP protocol, the maximum range for a DPC is 14 bits.		
:	Link Set ID The Link Set that carries traffic for this route (0x00 – 0x1F). If the DPC field is not 0xFFFFFFFF, a value of 0xFF in this field indicates de-configuration of the specified route.		

SS7 Signaling Route Configure 0x005F

:	<p>Priority Signaling traffic is routed over available link sets according to the priority established by the host (0x00 is the lowest priority and 0x23 is the highest priority). Link sets with the same priority to a destination load share traffic as a combination link set.</p>
:	<p>Combined Linkset Reference Indicates that the route being configured and the route given in the reference are both part of a combined linkset.</p> <p>Data [0-1] Route ID is given as "Combined LinkSet Reference".</p> <p>Note: The route that has already been configured becomes the Combined LinkSet Reference. Also, if there are n routes already configured as a combined linkset to a destination, then the route for the same combined linkset can be configured by giving any of the n-route IDs as a Combined Linkset Reference.</p>
:	<p>Checksum</p>

SS7 Signaling Route Query 0x0066

SwitchKit Name SS7SignalingRouteQuery

Type EXS API and SwitchKit API message

Description **SS7 Signaling Route Query 0x0066**

This message retrieves the configuration and status of all indicated signaling routes. This message can also be used to query the Destination ID, given the Route ID.

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE StackID;
    unsigned short Destination;
    unsigned short RouteID;
} XL_SS7SignalingRouteQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE StackID;
    unsigned short Destination;
    unsigned short RouteID;
    int DPC;
    UBYTE LinkSetID;
    UBYTE Priority;
    UBYTE DestinationStatus;
    UBYTE RouteStatus;
} XL_SS7SignalingRouteQueryAck;
```

C++ Class

```
class XLC_SS7SignalingRouteQuery : public
    XLC_OutboundMessage {
public:
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    unsigned short getDestination() const;
    void setDestination(unsigned short x);
    unsigned short getRouteID() const;
    void setRouteID(unsigned short x);
};
```

C++ Class Response

```
class XLC_SS7SignalingRouteQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
```

```

void setStatus(unsigned short x);
UBYTE getStackID() const;
void setStackID(UBYTE x);
unsigned short getDestination() const;
void setDestination(unsigned short x);
unsigned short getRouteID() const;
void setRouteID(unsigned short x);
int getDPC() const;
void setDPC(int x);
UBYTE getLinkSetID() const;
void setLinkSetID(UBYTE x);
UBYTE getPriority() const;
void setPriority(UBYTE x);
UBYTE getDestinationStatus() const;
void setDestinationStatus(UBYTE x);
UBYTE getRouteStatus() const;
void setRouteStatus(UBYTE x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x0066)	3, 4	Message Type (0x0066)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID

SS7 Signaling Route Query 0x0066

8 :	<u>AIB</u> Address Method 0x00 - Individual AEs	8, 9	Status MSB, LSB
	Number of AEs to follow AEs 0x20 SS7 Destination/Route <u>Stack ID</u> When you supply a valid Destination ID with the query, this field must contain the associated Stack ID. When the Destination ID supplied is 0xFFFF, the Stack ID can contain any SS7 Stack ID that is configured, as long as the Stack ID indicated exists on the SS7 card being queried. (The application might not know the correct Stack ID for this Route ID.) <u>Destination ID</u> 0x0000 to 0x003F - Valid 0x0040 to 0xFFFF - Invalid. The message gets a NACK of 0xEC, "Invalid Route Mode." <u>Route ID</u> 0x0000 - 0x01FF	10	0x20 SS7 Destination/Route <u>Stack ID</u> 0x00 - 0x03 If a valid Destination ID was supplied with the query, this field will contain the same stack ID in the response and the one supplied in the query. If the Destination ID supplied in the query was 0xFFFF, this field will contain the correct stack ID for the Route ID supplied in the query (i.e. the stack ID in the response will be the correct one, even though the one in the query may have been incorrect.) 0xFF- If the Destination ID supplied in the query was 0xFFFF, a response stack ID of 0xFF indicates that the Route ID supplied in the query is not configured and does not have an associated Stack ID. <u>Destination ID</u> 0x0000 to 0x003F - The Destination ID associated with the Route ID supplied in the query. 0xFFFF - If the Destination ID supplied in the query was 0xFFFF, a response Destination ID of 0xFFFF indicates that the Route ID supplied in the query is not configured and does not have an associated Destination ID. <u>Route ID</u> 0x0000 - 0x01FF Note: If it is the Destination ID that is being queried, a positive acknowledgment simply indicates that the Route ID is within range. In this case, a positive acknowledgment will be returned whether this Route ID is configured or not.
:	Checksum		
Response continued below.			

SS7 Signaling Route Query 0x0066

:	<p>DPC (four bytes) Every point code in the API messages will always be 4 bytes long with the first byte (MSB) as zero. See the SS7 section of the <i>Developer's Guide: CCS</i> for an explanation of how to convert the 4 bytes into ITU and ANSI point codes.</p> <p>0x00000000–0x00FFFFFF The DPC must be in this range for signaling stacks using 24-bit point codes (for example, ANSI).</p> <p>0x00000000–0x00003FFF The DPC must be in this range for signaling stacks using 14-bit point codes (for example, ITU).</p> <p>0xFFFFFFFF Indicates de-assignment of the specified destination and all routes to the specified destination. This is the only method that can be used to de-assign a destination.</p> <p>If using the BT IUP protocol, the maximum range for a DPC is 14 bits.</p>
:	<p>Link Set ID The Link Set that carries traffic for this route (0x00 – 0x1F). If the DPC field is not 0xFFFFFFFF, a value of 0xFF in this field indicates de-configuration of the specified route.</p>
:	<p>Priority Signaling traffic is routed over available link sets according to the priority established by the host (0x00 is the lowest priority and 0x23 is the highest priority). Link sets with the same priority to a destination load share traffic as a combination link set.</p>
:	<p>Destination Status</p> <p>0x00 - Inaccessible 0x01 - Accessible</p>
:	<p>Route Status</p> <p>0x00 - Unavailable 0x01 - Restricted 0x02 - Available</p>
:	<p>Combined Linkset Reference Indicates that the route being queried and the route given in the reference are both part of a combined linkset.</p> <p>LSB of the number of additional routes that are part of this destination/route ID. MSB of the number of additional routes that are part of this destination/route ID.</p> <p>LSB of Route ID MSB of Route ID</p> <p>Two bytes above repeat for additional Route IDs.</p>
:	<p>Checksum</p>

SS7 Signaling Stack Configure 0x005C

SwitchKit Name SS7SignalingStackConfig

Type EXS API and SwitchKit API message

Description **SS7 Signaling Stack Configure 0x005C**

Use this message to define a signaling stack, which includes assigning an OPC and a variant. You can also use this message to deconfigure a signaling stack.

NOTE: PCM encoding for CICs controlled by the signaling stack is defaulted based on the network signaling variant. ANSI stacks default to u-law regardless of the physical link type. ITU stacks default to A-law. To change the encoding format see the PCM Encoding Format message.

Sent by: Host

SwitchKit Code **Configuration**

```
SS7SignalingStackConfig (
    Node = integer,
    Slot = integer,
    Stack = integer,
    OPC = integer,
    NumModules = integer,
    Data = byte array);
```

C Structure

```
typedef struct {
    UBYTE Slot;
    UBYTE Stack;
    int OPC;
    UBYTE NumModules;
    UBYTE Data[218];
} XL_SS7SignalingStackConfig;
```

C++ Class

```
class XLC_SS7SignalingStackConfig : public
    XLC_OutboundMessage {
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getStack() const;
    void setStack(UBYTE x);
    int getOPC() const;
    void setOPC(int x);
    UBYTE getNumModules() const;
    void setNumModules(UBYTE x);
    const UBYTE *getData() const;
```

```
UBYTE *getData();
void setData(UBYTE *x);
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x005C)	3, 4	Message Type (0x005C)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB)
:	Address Method 0x00 - Individual AEs		The meaning of these fields depends upon the value of the <i>Entity</i> field in the message.
			Product License Data The following value can be received: 0x007F Software module still locked. This value is returned when attempting to configure a locked module for which no product license key has been downloaded.
		10	Checksum
	Number of AEs to follow		
	AEs		
	0x21 SS7 Slot		
:	OPC (four bytes) The address assigned to the CSP by the network. Every point code in the API messages will always be 4 bytes long, with the first byte (MSB) as zero. See the SS7 section of the <i>API Developer's Guide: CCS</i> for an explanation of how to convert the 4 bytes into ITU and ANSI point codes. 0x00000000–0x00FFFFFF: The OPC must be in this range for signaling stacks using 24-bit point codes, for example ANSI. 0x00000000–0x00003FFF: The OPC must be in this range for signaling stacks using 14-bit point codes, for example ITU. If using the BT IUP protocol, the maximum range for OPC is 14 bits. 0xFFFFFFFF: This value indicates deconfiguration of the specified signaling stack. All link sets, links, destinations, and routes previously configured to this signaling stack must be deconfigured before you deconfigure the signaling stack.		
13	Number of Modules (n)		

11	<p>Module 1: Type</p> <p>0x01 MTP (Message Transfer Part) Required for any SS7 configuration or for BT IUP</p> <p>0x02 ISUP (ISDN User Part) Required if you are using the ISUP Call Control User Part</p> <p>0x03 L3P (Layer 3 Plus) Required for any SS7 configuration</p> <p>0x04 TUP (Telephone User Part) Required if you are using the TUP Call Control User Part</p> <p>0x05 L3P TUP/BT IUP (Layer 3 Plus for Telephone User Part/BT IUP) Required if you are using the TUP or BT IUP Call Control User Part</p> <p>0x06 SCCP</p> <p>0x07 TCAP</p> <p>0x08 M3UA</p> <p>When using BT IUP, you must configure MTP. For example, if you are configuring the ISUP Call Control User Part, then you would send this message with three sets of module <i>Type</i> and <i>Variant</i> fields: MTP, L3P, and ISUP.</p> <p>When using the BT IUP variant, the Module Variant is 0x02. You must also configure module types 0x01 and 0x03. For example, in an E1 configuration, you would specify Module Type 0x01 with a variant of 0x01, Module 3 with a variant of 0x01, Module 4 with a variant of 0x02, and Module 5 with a variant of 0x02.</p>
:	Module 1: Variant (see information below this table)
:	Module n: Type
:	Module n: Variant
:	Checksum

Module Variant The following variants are tested and supported.

Value	Description	Modules
0x00	ANSI '97	All Modules (except TUP or BT IUP)
0x01	ITU-TS '97	All Modules
0x02	BT IUP	Module Types 4 and 5 when BT IUP is required
0x03	SSUTR2	Module Types 4 and 5 when BT IUP is required
0x08	JT	Module Types 1, 2, and 3

SS7 Signaling Stack Query 0x006D

SwitchKit Name	SS7SignalingStackQuery
Type	EXS API and SwitchKit API message
Description	SS7 Signaling Stack Query 0x006D This message retrieves the configuration of an SS7 signaling stack.
Sent by	Host
SwitchKit Code	C Structure

```
typedef struct {
    UBYTE StackID;
} XL_SS7SignalingStackQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE StackID;
    int OPC;
    UBYTE NumModules;
    UBYTE Data[216];
} XL_SS7SignalingStackQueryAck;
```

C++ Class

```
class XLC_SS7SignalingStackQuery : public
    XLC_OutboundMessage {
public:
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
};
```

C++ Class Response

```
class XLC_SS7SignalingStackQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    int getOPC() const;
    void setOPC(int x);
    UBYTE getNumModules() const;
    void setNumModules(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x006D)	3, 4	Message Type (0x006D)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow	10	AIB Same as message.
	AEs 0x08 SS7 Stack	:	OPC (four bytes) Every point code in the API messages will always be 4 bytes long with the first byte (MSB) as zero. See the SS7 section of the <i>API Developer's Guide: CCS</i> for an explanation of how to convert the 4 bytes into ITU and ANSI point codes.
:	Checksum	:	Number of Modules (n)
Response continued below.			
:	Module 1: Type 0x01 MTP 0x02 ISUP 0x03 L3P 0x04 TUP 0x05 L3P TUP 0x06 SCCP 0x07 TCAP		
:	Module 1: Variant 0x00 ANSI 0x01 ITU-TS		
:	Module n: Type		
:	Module n: Variant		
:	Checksum		

SS7 TUP Message Format Configure 0x008F

SwitchKit Name	SS7TUPMessageFormatConfig
Type	EXS API and SwitchKit API message
Description	<p>SS7 TUP Message Format Configure 0x008F</p> <p>This message allows the host to modify the format of a TUP message in the TUP Message Configuration template. The template of each message is used to format outgoing messages and to interpret incoming messages.</p> <p>You can use up to a maximum of eight mandatory fields and 12 optional fields per message. You can use a maximum of eight subfields per field.</p> <p>Use the <i>SS7 TUP Message Query</i> message to retrieve the currently configured format of a TUP message.</p> <p>Modifying the format of an TUP message may also require modification of its PPL Configuration Bytes.</p>
Sent by	Host

SwitchKit Code **Configuration**

```
SS7TUPMessageFormatConfig (
    Node = integer,
    StackID = integer,
    TUPMessageIndex = integer,
    MCTEntry = integer,
    H0Length = integer,
    H0Value = integer,
    H1Length = integer,
    H1Value = integer,
    MsgAttributes = integer,
    Data = byte array);
```

C Structure

```
typedef struct {
    UBYTE StackID;
    UBYTE TUPMessageIndex;
    UBYTE MCTEntry;
    UBYTE H0Length;
    UBYTE H0Value;
    UBYTE H1Length;
    UBYTE H1Value;
    unsigned short MsgAttributes;
    UBYTE Data[215];
} XL_SS7TUPMessageFormatConfig;
```

C++ Class

```
class XLC_SS7TUPMessageFormatConfig : public
    XLC_OutboundMessage {
public:
```

```

UBYTE getStackID() const;
void setStackID(UBYTE x);
UBYTE getTUPMessageIndex() const;
void setTUPMessageIndex(UBYTE x);
UBYTE getMCTEntry() const;
void setMCTEntry(UBYTE x);
UBYTE getH0Length() const;
void setH0Length(UBYTE x);
UBYTE getH0Value() const;
void setH0Value(UBYTE x);
UBYTE getH1Length() const;
void setH1Length(UBYTE x);
UBYTE getH1Value() const;
void setH1Value(UBYTE x);
unsigned short getMsgAttributes() const;
void setMsgAttributes(unsigned short x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};
    
```

EXS API Hex Format

NOTE: The fields listed between the double lines in the table (*Field[m] ñ ID through Subfield[s] ñ Maximum Length, LSB*) are repeated for each mandatory field. Likewise, the fields listed between the thick lines in the table (*Field[o] ñ ID through Subfield[s] ñ Maximum Length, LSB*) are repeated for each optional field. Each mandatory and optional field can have a variable number of subfields. For example, Mandatory Field[0] can have 3 subfields, Mandatory Field[1] can have two subfields, and Optional Field[0] can have 4 subfields.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x008F)	3, 4	Message Type (0x008F)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8, 9	Status MSB, LSB
:	Address Method 0x00 - Individual AEs	10	Checksum
	Number of AEs to follow		
	AEs 0x08 SS7 Stack		
:	TUP Message Index		
:	MCT Entry Indicates whether the entry in the Message Configuration Table is valid or invalid. 0x00 Invalid (will be ignored) 0x01 Valid (will be handled by TUP)		

SS7 TUP Message Format Configure 0x008F

:	H0 Length
:	H0 Value
:	H1 Length
:	H1 Value
:	<p>Message Attributes MSB, LSB This field is a bit mask. The meaning of each bit is described below.</p> <p>Bit 0 Group Message Flag 0 Not Group Message 1 Group Message</p> <p>Bits 1–15 Reserved, must be 0x00</p>
:	Number of Mandatory Fields (M)
:	Field[m] – ID
:	<p>Field[m] – Attributes MSB, LSB This field is a bit mask. The meaning of each bit is described below.</p> <p>Bit 0 Range and Status Field Flag 0 Field is not Range and Status Field 1 Field is Range and Status Field</p> <p>Bit 1 Filler Field Flag 0 Field is not Filler Field 1 Field is Filler Field (not reported to or required from the host)</p> <p>Bit 2 Status Subfield Variant 0 Field is not a Range and Status field or is a Range and Status field and the status bits are used to inform TUP as to which CICs are being operated (such as the group blocking messages) 1 Field is a Range and Status field and all CICs in the range are being operated, independent of the status value (such as, the GRA message)</p> <p>Bit 3 Zero length subfield allowed 0 If present in the SSUTR2 field, access field with 0 length will not be automatically inserted into the message. 1 If present in the SSUTR2 field, access field with 0 length will be automatically inserted into the message.</p>

	<p>Bit 4 Indicator Field Flag</p> <p>0 Indicator will not be inserted.</p> <p>1 If present in the SSUTR2 field, an indicator will be automatically inserted. If this indicator bit is set for optional field, all the optional parameters followed by the indicator will belong to this indicator. If this indicator is set for mandatory field, all the fields with an extension field indicator will belong to the indicator.</p> <p>Bits 5-12 Reserved Bit Mask.</p> <p>Bit Mask is used to set bits in the indicator field. This allows the field location to be anywhere in the indicator. They need not be in order.</p> <p>Bit 13 Common Field Flag</p> <p>0 Not a common field</p> <p>1 This field is a common optional field. Its presence is dependent on certain other fields following it. Its presence does not cause a bit to be set in the indicator field.</p> <p>Bit 14 Extension Field Flag</p> <p>0 Not an extension field</p> <p>1 This field is an extension field of the previous field and will cause a bit to be set in the indicator.</p> <p>Bit 15 Reserved; must be 0x00</p>
:	Field[m] – Number of Subfields [S]
:	Subfield[s] – ID
:	<p>Subfield[s] – Attributes MSB, LSB</p> <p>This field is a bit mask. The meaning of each bit is described below.</p> <p>Bits 0–2 Number of Units to Add to Subfield if Length Subfield (0–7)</p> <p>Bits 3–4 Subfield Length Type if Length Subfield</p> <p>0 Bits</p> <p>1 Half-Octets</p> <p>2 Octets</p> <p>Bit 5 Subfield Type Flag</p> <p>0 Data Subfield (Bits 0–4 are ignored)</p> <p>1 Length Subfield (The value of the Subfield indicates the length of the next Data Subfield)</p> <p>Bit 6 Use Filler (Pad subfield to the next octet)</p> <p>Bit 7 Is Filler (Subfield is a filler/does not contain useful data)</p> <p>Bits 8–15 Reserved, must be 0x00</p>
:	Subfield[s] – Min. Length MSB, LSB
:	Subfield[s] – Max. Length MSB, LSB

:	Optional Part – Attributes MSB, LSB This field is a bit mask. The meaning of each bit is described below. Bit 0 Optional Field Use Filler Flag 0 If present in the TUP message, the Optional Field Indicator bitfield will not be octet-padded. 1 If present in the TUP message, the Optional Field Indicator bitfield will be octet-padded. Bits 1–8 Bit Mask is used to set bits in the indicator field. This allows the optional field location anywhere in the indicator. They need not be in order. Bits 9-12 Length of optional indicator to be inserted Bits 13-15 Reserved; must be 0x00
:	Number of Optional Fields (0)
:	Field[0] – ID
:	Field[0] – Attributes MSB, LSB
:	Field[0] – Number of Subfields [S]
:	Subfield[s] – ID
:	Subfield[s] – Attributes MSB, LSB
:	Subfield[s] – Min. Length MSB, LSB
:	Subfield[s] – Max. Length MSB, LSB
:	Checksum

TUP Message Index

This field indicates the index into the TUP Message Format table for the specified TUP message. For more information on the TUP Message Format table, refer to the *API Developer’s Guide: Common Channel Signaling*.

Inde	Messag	Inde	Messag	Index	Message
0x00	IAM	0x12	ACC	0x24	BLO
0x01	IAI	0x13	ADI	0x25	BLA
0x02	SAM	0x14	CGC	0x26	UBL
0x03	SAO	0x15	DPN	0x27	UBA
0x04	ACM	0x16	LOS	0x28	MGB
0x05	ANC	0x17	NNC	0x29	MBA
0x06	ANN	0x18	SEC	0x2A	MGU
0x07	ANU	0x19	SSB	0x2B	MUA
0x08	CBK	0x1	SST	0x2C	HGB
0x09	CLF	0x1B	MPR	0x2D	HBA
0x0A	RLG	0x1C	EUM	0x2E	HGU
0x0B	GRQ	0x1D	UNN	0x2F	HUA
0x0C	GSM	0x1E	CCR	0x30	SGB
0x0D	RAN	0x1F	COT	0x31	SBA
0x0E	FOT	0x20	CCF	0x32	SGU
0x0F	CCL	0x21	GRS	0x33	SUA

Inde	Messag	Inde	Messag	Index	Message
0x10	CFL	0x22	GRA	0x34–0x43	Reserved for future use
0x11	ACB	0x23	RSC	0x44–0x5B	User-defined Messages

SSUTR2 Message Index

This field indicates the index into the SSUTR2 Message Format table for the specified SSUTR2 message. For more information on the SSUTR2 Message Format table, refer to the *API Developer's Guide: Common Channel Signaling*.

0x00	MIF	0x02	MSA
0x03	MSS	0x04	ACF
0x05	RIU	0x08	RAU
0x09	FIU	0x0A	LIG
0x0B	DEG	0x0C	IFG
0x0D	NRP	0x10	ECH
0x11	ACI	0x14	EFC
0x16	LHS	0x17	ERN
0x18	EEC	0x19	OCC
0x1A	TSI	0x1B	INU
0x1D	NNU	0x1E	CCD
0x1F	CCP	0x20	CCN
0x21	GRS	0x22	GRA
0x23	RZC	0x24	BLO
0x25	BLA	0x26	UBL
0x27	UBA	0x34	CHT
0x35	ITX	0x36	TXA
0x37	TAX	0x38	SND
0x39	EAR	0x3A	MCE
0x3B	MUU		

H0/H1 Length

The system uses the H0 and H1 fields as *x, y* coordinates for the Heading Code Allocations table found in the Q.723 specification.

This field indicates whether length of the *H0 Value* and *H1 Value* fields are defined as half-octets or octets (groups of 4 bits or 8 bits, respectively). Valid entries for this field are as follows:

- 0x01 Half-octets
- 0x02 Octets

H0/H1 Value

The system uses the H0 and H1 fields as *x, y* coordinates for the Heading Code Allocations table found in the Q.723 specification.

TUP This field represents the value of the H0 or H1 parameter in the Heading Code Allocations table. To represent the appropriate Heading Code, use the following H0 and H1 values:

Heading	H0	H1
IAM	0x01	0x01
IAI	0x01	0x02
SAM	0x01	0x03
SAO	0x01	0x04
GSM	0x02	0x01
COT	0x02	0x03
CCF	0x02	0x04
GRQ	0x03	0x01
ACM	0x04	0x01
CHG	0x04	0x02
SEC	0x05	0x01
CGC	0x05	0x02
NNC	0x05	0x03
ADI	0x05	0x04
CFL	0x05	0x05
SSB	0x05	0x06
UNN	0x05	0x07
LOS	0x05	0x08
SST	0x05	0x09
ACB	0x05	0x0A
DPN	0x05	0x0B
MPR	0x05	0x0C
EUM	0x05	0x0F
ANU	0x06	0x00
ANC	0x06	0x01
ANN	0x06	0x02
CBK	0x06	0x03
CLF	0x06	0x04
RAN	0x06	0x05
FOT	0x06	0x06
CCL	0x06	0x07
RLG	0x07	0x01
BLO	0x07	0x02
BLA	0x07	0x03
UBL	0x07	0x04
UBA	0x07	0x05
CCR	0x07	0x06
RSC	0x07	0x07
MGB	0x08	0x01
MBA	0x08	0x02
MGU	0x08	0x03
MUA	0x08	0x04

Heading	H0	H1
HGB	0x08	0x05
HBA	0x08	0x06
HGU	0x08	0x07
HUA	0x08	0x08
GRS	0x08	0x09
GRA	0x08	0x0A
SGB	0x08	0x0B
SBA	0x08	0x0C
SGU	0x08	0x0D
SUA	0x08	0x0E
ACC	0x0A	0x01

SSUTR2 This field represents the value of the H0 or H1 parameter in the Heading Code Allocations table. To represent the appropriate Heading Code, use the following H0 and H1 values:

Heading	H0	H1
CHT	0x00	0x03
ITX	0x00	0x04
TXA	0x00	0x08
MSA	0x01	0x03
MSS	0x01	0x04
IFG	0x02	0x02
CCP	0x02	0x03
CCN	0x02	0x04
DEG	0x03	0x01
TAX	0x04	0x02
EEC	0x05	0x01
EFC	0x05	0x02
ERN	0x05	0x03
ECH	0x05	0x05
OCC	0x05	0x06
NNU	0x05	0x07
LHS	0x05	0x08
TSI	0x05	0x09
ACI	0x05	0x0A
INU	0x05	0x0C
NRP	0x06	0x05
LIG	0x07	0x01
BLO	0x07	0x02
BLA	0x07	0x03
UBL	0x07	0x04
UBA	0x07	0x05
CCD	0x07	0x06
RZC	0x07	0x07
GRS	0x08	0x09
GRA	0x08	0x0A
MUU	0x0B	0x01

Heading	H0	H1
MCE	0x0B	0x02
ACF	0x0C	0x02
MIF	0x0D	0x02
SND	0x0E	0x01
EAR	0x0E	0x02
RIU	0x0F	0x01
RAU	0x0F	0x02
FIU	0x0F	0x03

SSUTR2 Heading Allocation Table

The actual SSUTR2 Heading Allocations Table is provided here for reference:

H0 / H1	0000	0001	0010	0011	0100	0101	0110	0111
0000				CHT	ITX			
0001				MSA	MSS			
0010			IFG	CCP	CCN			
0011		DEG						
0100			TAX					
0101		EEC	EFC	ERN		ECH	OCC	NNU
0110						NRP		
0111		LIG	BLO	BLA	UBL	UBA	CCD	RZC
1000								
1001	Resv.							
1010	Resv.							
1011		MUU	MCE					
1100			ACF					
1101			MIF					
1110		SND	EAR					
1111		RIU	RAU	FIU				

H0 / H1	1000	1001	1010	1011	1100
0000	TXA				
0001					
0010					
0011					
0100					
0101	LHS	TSI	ACI		INU
0110					
0111					
1000		GRS	GRA		
1001	Resv.				
1010					
1011					
1100					
1101					
1110					

H0 / H1	1000	1001	1010	1011	1100
1111					

Field ID The table below shows the Field ID for supported fields.

Field ID	Field Name	TUP Messages Containing Field
0x01	Calling Party Category	IAM, IAI, GSM
0x02	Call Setup Message Indicators	IAM, IAI
0x03	Called Party Address Data	IAM, IAI, SAM
0x04	IAI National Use Field	IAI
0x05	Closed User Group Information	IAI
0x06	Additional Calling Party Information	IAI
0x07	Additional Routing Information	IAI
0x08	Calling Line ID	IAI, GSM
0x09	Original Called Address	IAI, GSM
0x0A	Charging Information	IAI
0x0B	Called Party Address Digit	SAO
0x0C	ACM Message Indicator	ACM
0x0D	Request Type Indicator	GRQ
0x0E	RESERVED	RESERVED
0x0F	Incoming Trunk and Transit Identity	GSM
0x10	Echo Suppressor Indicator *	GSM
0x11	Malicious Call Identification *	GSM
0x12	Hold Indicator *	GSM
0x13	ACC Message Indicator	ACC
0x14	EUM Octet Indicator	EUM
0x15	Signaling Point Code	EUM
0x16	Range and Status Field	GRA, MGB, MBA, MGU, MUA, HGB, HBA, HGU, HUA, SGB, SBA, SGU, SUA
0x17	Range Field (w/ no status)	GRS

* These are only flags in the Indicator field and contain no data (Data Length = 0)

SSUTR2

Field ID	Field Name	SSUTR2 Messages containing Field
0x01	Calling Party Category	MIF, IFG
0x02	Call Setup Message Indicators	MIF
0x03	Called Party Address Data	MIF, MSA
0x04-0x05	-----	-----

Field ID	Field Name	SSUTR2 Messages containing Field
0x06	Additional Calling Party Information	MIF
0x07	Additional Routing Information	MIF
0x08	Calling Line ID	MIF, IFG
0x09-0x0A	-----	-----
0x0B	Called Party Address Digit	MSS
0x0C	ACF Message Indicator	ACF
0x0D	Request Type Indicator	DEG
0x0E-0x17	-----	-----
0x18	Type of Access Requested	ACF
0x19	Indicator of presence of data	ACF
0x1A	Amount of Price and number of call charge units A	TAX
0x1B	Causes	EAR, SND
0x1C	Access Information Field w zero length allowed	EAR, RAN, CLB, CLF, ACF,RIU
0x1D	Access Information Field	MUU, MCE, MIF
0x1E	Pricing factor	CHT, TAX
0x1F	Time Indicator	CHT, TAX
0x20	Price Allocation Domain	ITX
0x21	Message Number	ITX
0x22	Price Indicator	CHT
0x23	Amount of Price and number of call charge units B	TAX
0x24	Identity of first called party	MIF
0x25	Pricing factor B	TAX

SS7 TUP Message Query 0x0090

SwitchKit Name	SS7TUPMessageQuery
Type	EXS API and SwitchKit API message
Description	SS7 TUP Message Query 0x0090 This message reports the configured format of the specified TUP message.
Sent by	Host
SwitchKit Code	C Structure

```
typedef struct {
    UBYTE StackID;
    UBYTE TUPMessageIndex;
} XL_SS7TUPMessageQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE StackID;
    UBYTE TUPMessageIndex;
    UBYTE MCTEntry;
    UBYTE H0Length;
    UBYTE H0Value;
    UBYTE H1Length;
    UBYTE H1Value;
    unsigned short MsgAttributes;
    UBYTE Data[213];
} XL_SS7TUPMessageQueryAck;
```

C++ Class

```
class XLC_SS7TUPMessageQuery : public XLC_OutboundMessage
{
public:
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    UBYTE getTUPMessageIndex() const;
    void setTUPMessageIndex(UBYTE x);
};
```

C++ Class Response

```
class XLC_SS7TUPMessageQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getStackID() const;
    void setStackID(UBYTE x);
    UBYTE getTUPMessageIndex() const;
    void setTUPMessageIndex(UBYTE x);
```

```

UBYTE getMCTEntry() const;
void setMCTEntry(UBYTE x);
UBYTE getH0Length() const;
void setH0Length(UBYTE x);
UBYTE getH0Value() const;
void setH0Value(UBYTE x);
UBYTE getH1Length() const;
void setH1Length(UBYTE x);
UBYTE getH1Value() const;
void setH1Value(UBYTE x);
unsigned short getMsgAttributes() const;
void setMsgAttributes(unsigned short x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};
    
```

EXS API Hex Format

NOTE: The fields listed between (*Field[m] – ID* through *Subfield[s] – Maximum Length, LSB*) are repeated for each mandatory field. Likewise, the fields listed between (*Field[o] – ID* through *Subfield[s] – Maximum Length, LSB*) are repeated for each optional field.

Each mandatory and optional field can have a variable number of subfields. For example, Mandatory Field[0] can have 3 subfields, Mandatory Field[1] can have two subfields, and Optional Field[0] can have 4 subfields.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0090)	3, 4	Message Type (0x0090)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow	10	AIB (same as message)
	AE 0x08 SS7 Stack		TUP Message Index (See information below this table)
:	TUP Message Index	:	MCT Entry
:	Checksum		Indicates whether the entry in the Message Configuration Table is valid or invalid. 0x00 Invalid (will be ignored) 0x01 Valid (to be handled by TUP)
	Response continued below.		
:	H0 Length (See information below this table)		

SS7 TUP Message Query 0x0090

:	H0 Value (See information below this table)
:	H1 Length (See information below this table)
:	H1 Value (See information below this table)
:	<p>Message Attributes MSB, LSB This field is a bit mask. The meaning of each bit is described below.</p> <p>Bit 0 Group Message Flag 0 Not Group Message 1 Group Message</p> <p>Bits 1–15 Reserved, must be 0x00</p>
:	Number of Mandatory Fields (M)
:	Field[m] – ID (See information below this table)
:	<p>Field[m] – Attributes MSB, LSB This field is a bit mask. The meaning of each bit is described below.</p> <p>Bit 0 Range and Status Field Flag 0 Field is not Range and Status Field 1 Field is Range and Status Field</p> <p>Bit 1 Filler Field Flag 0 Field is not Filler Field 1 Field is Filler Field (not reported to or required from the host)</p> <p>Bit 2 Status Subfield Variant 0 Field is not a Range and Status field or is a Range and Status field and the status bits are used to inform TUP as to which CICs are being operated (such as the group blocking messages) 1 Field is a Range and Status field and all CICs in the range are being operated, independent of the status value (such as the GRA message)</p> <p>Bits 3-15 Reserved, must be 0x00</p>
:	Field[m] – Number of Subfields [S]
:	Subfield[s] – ID

:	Subfield[s] – Attributes MSB, LSB This field is a bit mask. The meaning of each bit is described below. Bits 0-2 Number of Units to Add to Subfield if Length Subfield (0–7) Bits 3-4 Subfield Length Type if Length Subfield 0 Bits 1 Half-Octets 2 Octets Bit 5 Subfield Type Flag 0 Data Subfield (Bits 0–4 are ignored) 1 Length Subfield (The value of the Subfield indicates the length of the next Data Subfield) Bit 6 Use Filler (Pad subfield to the next octet) Bit 7 Is Filler (Subfield is a filler/does not contain useful data) Bits 8–15 Reserved, must be 0x00
:	Subfield[s] – Minimum Length MSB, LSB
:	Subfield[s] – Maximum Length MSB, LSB
:	Optional Part – Attributes MSB, LSB This field is a bit mask. The meaning of each bit is described below. Bit 0 Optional Field Use Filler Flag 0 If present in the TUP message, the Optional Field Indicator bitfield will not be octet-padded. 1 If present in the TUP message, the Optional Field Indicator bitfield will be octet-padded. Bits 1–15 Reserved, must be 0x00
:	Number of Optional Fields (O)
:	Field[0] – ID
:	Field[0] – Attributes MSB, LSB
:	Field[0] – Number of Subfields [S]
:	Subfield[s] – ID
:	Subfield[s] – Attributes MSB, LSB
:	Subfield[s] – Minimum Length MSB, LSB
:	Checksum

TUP Message Index This field indicates the index into the TUP Message Format table for the specified TUP message. For more information on the TUP Message Format table, refer to the *CSP Developer’s Guide: Common Channel Signaling*.

Index	Message	Index	Message	Index	Message
0x00	IAM	0x12	ACC	0x24	BLO
0x01	IAI	0x13	ADI	0x25	BLA
0x02	SAM	0x14	CGC	0x26	UBL
0x03	SAO	0x15	DPN	0x27	UBA

Index	Message	Index	Message	Index	Message
0x04	ACM	0x16	LOS	0x28	MGB
0x05	ANC	0x17	NNC	0x29	MBA
0x06	ANN	0x18	SEC	0x2A	MGU
0x07	ANU	0x19	SSB	0x2B	MUA
0x08	CBK	0x1A	SST	0x2C	HGB
0x09	CLF	0x1B	MPR	0x2D	HBA
0x0A	RLG	0x1C	EUM	0x2E	HGU
0x0B	GRQ	0x1D	UNN	0x2F	HUA
0x0C	GSM	0x1E	CCR	0x30	SGB
0x0D	RAN	0x1F	COT	0x31	SBA
0x0E	FOT	0x20	CCF	0x32	SGU
0x0F	CCL	0x21	GRS	0x33	SUA
0x10	CFL	0x22	GRA	0x34–0x43	Reserved for future use
0x11	ACB	0x23	RSC	0x44–0x5B	User-defined Messages

H0/H1 Length

The system uses the H0 and H1 fields as *x, y* coordinates for the Heading Code Allocations table found in the Q.723 specification.

This field indicates whether length of the *H0 Value* and *H1 Value* fields are defined as half-octets or octets (groups of 4 bits or 8 bits, respectively). Valid entries for this field are as follows:

- 0x01 Half-octets
- 0x02 Octets

H0/H1 Value

The system uses the H0 and H1 fields as *x, y* coordinates for the Heading Code Allocations table found in the Q.723 specification.

This field represents the value of the H0 or H1 parameter in the Heading Code Allocations table. To represent the appropriate Heading Code, use the following H0 and H1 values:

Heading	H0	H1
IAM	0x01	0x01
IAI	0x01	0x02
SAM	0x01	0x03
SAO	0x01	0x04
GSM	0x02	0x01
COT	0x02	0x03
CCF	0x02	0x04
GRQ	0x03	0x01
ACM	0x04	0x01
CHG	0x04	0x02
SEC	0x05	0x01

Heading	H0	H1
CGC	0x05	0x02
NNC	0x05	0x03
ADI	0x05	0x04
CFL	0x05	0x05
SSB	0x05	0x06
UNN	0x05	0x07
LOS	0x05	0x08
SST	0x05	0x09
ACB	0x05	0x0A
DPN	0x05	0x0B
MPR	0x05	0x0C
EUM	0x05	0x0F
ANU	0x06	0x00
ANC	0x06	0x01
ANN	0x06	0x02
CBK	0x06	0x03
CLF	0x06	0x04
RAN	0x06	0x05
FOT	0x06	0x06
CCL	0x06	0x07
RLG	0x07	0x01
BLO	0x07	0x02
BLA	0x07	0x03
UBL	0x07	0x04
UBA	0x07	0x05
CCR	0x07	0x06
RSC	0x07	0x07
MGB	0x08	0x01
MBA	0x08	0x02
MGU	0x08	0x03
MUA	0x08	0x04
HGB	0x08	0x05
HBA	0x08	0x06
HGU	0x08	0x07
HUA	0x08	0x08
GRS	0x08	0x09
GRA	0x08	0x0A
SGB	0x08	0x0B
SBA	0x08	0x0C
SGU	0x08	0x0D
SUA	0x08	0x0E
ACC	0x0A	0x01

Field ID The table below shows the Field ID for supported fields.

Field ID	Field Name	TUP Messages Containing Field
0x01	Calling Party Category	IAM, IAI, GSM
0x02	Call Setup Message Indicators	IAM, IAI
0x03	Called Party Address Data	IAM, IAI, SAM
0x04	IAI National Use Field	IAI
0x05	Closed User Group Information	IAI
0x06	Additional Calling Party Information	IAI
0x07	Additional Routing Information	IAI
0x08	Calling Line ID	IAI, GSM
0x09	Original Called Address	IAI, GSM
0x0A	Charging Information	IAI
0x0B	Called Party Address Digit	SAO
0x0C	ACM Message Indicator	ACM
0x0D	Request Type Indicator	GRQ
0x0E	RESERVED	RESERVED
0x0F	Incoming Trunk and Transit Identity	GSM
0x10	Echo Suppressor Indicator *	GSM
0x11	Malicious Call Identification *	GSM
0x12	Hold Indicator *	GSM
0x13	ACC Message Indicator	ACC
0x14	EUM Octet Indicator	EUM
0x15	Signaling Point Code	EUM
0x16	Range and Status Field	GRA, MGB, MBA, MGU, MUA, HGB, HBA, HGU, HUA, SGB, SBA, SGU, SUA
0x17	Range Field (w/ no status)	GRS

* These are only flags in the Indicator field and contain no data (Data Length = 0)

Standby Line Card Configure 0x0023

SwitchKit Name Standby Line Card Config

Type EXS API and SwitchKit API message

Description **Standby Line Card Configure 0x0023**

This message allows the host in a redundant system to configure a line card as a standby card. Only line cards that have Standby IO cards behind them can be designated as standby line cards. A *Card Status Report* message first informs the host which line card has a Standby I/O installed. The host must assign a line card as a standby before a switchover can occur.

Sent by Host

Example Message (Socket Log Output for SwitchKit)

In the following socket log output/example message, the host configures a T-ONE card to be a standby card:

```
00 0C 00 23 00 00 FF 00 01 01 01 0F 00 00
```

SwitchKit Code **Configuration**

```
StandbyLineCardConfig (
    Node = integer,
    Slot = integer,
    CardType = integer,
    Action = integer);
```

C Structure

```
typedef struct {
    UBYTE Slot;
    UBYTE CardType;
    UBYTE Action;
} XL_StandbyLineCardConfig;
```

C++ Class

```
class XLC_StandbyLineCardConfig : public XLC_SlotMessage
{
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getCardType() const;
    void setCardType(UBYTE x);
    UBYTE getAction() const;
    void setAction(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0023)	3, 4	Message Type (0x0023)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB)
:	Address MEthod	10	Checksum
	0x00 - Individual AEs		
	Number of AEs to follow		
	AEs		
	0x01 Slot		
:	Card Type		
	0x00 T-ONE or ST1LC		
	0x01 E-ONE or SE1LC		
	0x02 J-ONE or SJ1LC		
	0x03 DS3		
:	Action		
	0x00 Assign		
	0x01 De-assign		
:	Checksum		

Start Dial Configure 0x0013

SwitchKit Name	StartDialConfig
Type	EXS API and SwitchKit API message
Description	Start Dial Configure 0x0013 This message allows the host to change the default start dial types.
Sent by	Host
SwitchKit Code	Configuration

```
StartDialConfig (  
    Node = integer,  
    Range = StartSpan:StartChan - EndSpan:EndChan,  
    StartDialType = integer,  
    StartDialValue = integer);
```

C Structure

```
typedef struct {  
    unsigned short StartSpan;  
    UBYTE StartChannel;  
    unsigned short EndSpan;  
    UBYTE EndChannel;  
    UBYTE StartDialType;  
    UBYTE StartDialValue;  
} XL_StartDialConfig;
```

C++ Class

```
class XLC_StartDialConfig : public XLC_ChanRangeMessage {  
public:  
    unsigned short getStartSpan() const;  
    void setStartSpan(unsigned short x);  
    UBYTE getStartChannel() const;  
    void setStartChannel(UBYTE x);  
    unsigned short getEndSpan() const;  
    void setEndSpan(unsigned short x);  
    UBYTE getEndChannel() const;  
    void setEndChannel(UBYTE x);  
    UBYTE getStartDialType() const;  
    void setStartDialType(UBYTE x);  
    UBYTE getStartDialValue() const;  
    void setStartDialValue(UBYTE x);  
};
```


Statistics Query 0x0121

SwitchKit Name Statistics Query

Type EXS API and SwitchKit API message

Description **Statistics Query 0x0121**

NOTE: This message applies to the DSP Series 2 card only.

Use this message to query cache and function usage statistics that are kept on the DSP Series 2 card. The statistics are stored in cache for 16 days, and then they are replaced one-by-one with more recent statistics. At any time, you can view statistics for the past 16 days. You can configure the time to copy statistics into the buffer from one second to 86,400 seconds, in one-second increments.

This message does not support wildcards for the DSP Chip AIB, because the data that comes back from the CSP would be too long.

Sent by Host Application

SwitchKit Code **C Structure**

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE TLVCount;
    UBYTE Data[250];
} XL_StatisticsQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved[13];
    UBYTE AddrInfo[251];
    UBYTE TLVCount;
    UBYTE Data[250];
} XL_StatisticsQueryAck;
```

C++ Class

```
class XLC_StatisticsQuery : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);

    // Extended addressing functions
    // DSP Chip AIB functions
    void DSPChip(UBYTE x);
    UBYTE getDSPSlot() const;
```

```
void setDSPSlot(UBYTE x);
UBYTE getDPSIMM() const;
void setDPSIMM(UBYTE x);
// Slot AIB functions
UBYTE getSlot() const;
void setSlot(UBYTE x);
UBYTE getDataType() const;
void setDataType(UBYTE x);
UBYTE getTLVCount() const;
void setTLVCount(UBYTE x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x)
};
```

C++ Class Response

```
class XLC_StatisticsQueryAck : public XLC_OutboundMessage
{
public:

    unsigned short getStatus() const
    void setStatus(unsigned short x)
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x)
    ;
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x0121)	3, 4	Message Type (0x0121)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID

Statistics Query 0x0121

<p>8 :</p>	<p><u>AIB</u> Address Method 0x00 - Individual AEs</p>	<p>8, 9</p>	<p>Status (MSB, LSB)</p> <p>0x0001 - Invalid TLV Data Software can't find the TLV Data Buffer. This can also occur if the Data for a TLV is out of range.</p> <p>0x0003 - Invalid number of TLVs. 0x0004 - Invalid TLV Length 0x0006 - Invalid TLV 0x000D - Mandatory TLVs missing</p> <p>Also see Common Response Status Values</p>
	<p>Number of AEs to follow</p>	<p>10</p>	<p>Number of TLVs to Follow</p>
	<p>AE</p> <p>0x01 Slot (to query a DSP 2 card, or a CSP Matrix Series 3 Card for a system query)</p> <p>0x01 Slot and 0x22 DSP Chip (to query an individual DSP chip)</p> <p>NOTE: You can also do a system query by sending a Null AIB: Address Method = 0x00 Number of AEs = 0x00 No AE Data</p>	<p>:</p>	<p>TLVs</p> <p>The first TLV returned is the same Query Statistics TLV sent down by the host, indicating the object queried. 0x05F9 Query Statistics</p> <p>Additional TLVs returned depend on the entity queried.</p> <p>Query Type: 0x0400 - DSP Function Statistics 0x05F7 DSP Function Statistics</p> <p>Query Type: 0x0401 - NFS Statistics 0x05F8 NFS Statistics</p> <p>Query Type: 0x0402 - Fixed Memory Statistics 0x05F5 Fixed Memory Statistics</p> <p>Query Type: 0x0403 - DSP Series 2 Cache Statistics 0x05F6 DSP Series 2 Cache Statistics</p> <p>Query Type: 0x0404) - Resource Point Statistics 0x05FE Resource Point Statistics</p> <p>Query Type: 0x0405) - CPU Statistics 0x05FD CPU Statistics</p> <p>Query Type: 0x0406) - DSP 2 Overload Statistics 0x0616 DSP Series 2 Overload Statistics</p>
<p>:</p>	<p>Data Type 0x00 TLVs</p>	<p>:</p>	<p>Checksum</p>

Statistics Query 0x0121

:	Number of TLVs to Follow		
:	TLV 0x05F9 Query Statistics <u>Query Types</u> 0x0400 - DSP Function Statistics 0x0401 - NFS Statistics 0x0402 - Fixed Memory Statistics 0x0403 - DSP Series 2 Cache Statistics 0x0404 - Resource Point Statistics* 0x0405 - CPU Statistics 0x0406 - DSP 2 Overload Statistics 0xFFFF - All valid queries for the Slot or SIMM		
:	Checksum		

* TLV 0x05F9, query type 0404 is only valid while addressing the CSP Matrix Series 3 Card slot to query Resource Points. It does not work, if you address the DSP Series 2 card slot to query Resource Points.

Subrate Connection Management 0x000D

SwitchKit Name	SubrateConnectionManagement
Type	EXS API and SwitchKit API message
Description	<p>Subrate Connection Management 0x000D</p> <p>This message is used to establish and tear down subrate connections. Both channels should be out of service before the connection is established.</p>
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {
    BaseFields Base;
    unsigned short SubrateSpanA;
    unsigned short SubrateChannelA;
    UBYTE NumBitsA;
    unsigned short SubrateSpanB;
    unsigned short SubrateChannelB;
    UBYTE NumBitsB;
    UBYTE reserved27[20];
    UBYTE Action;
    UBYTE reserved48[222];
} XL_SubrateConnectionManagement;
```

C++ Class

```
class XLC_SubrateConnectionManagement : public
    XLC_OutboundMessage {
public:
    virtual MsgStruct *getStructPtr();
    virtual const MsgStruct *getStructPtr() const;
    virtual int getTag() const;
    unsigned short getSubrateSpanA() const;
    void setSubrateSpanA(unsigned short x);
    unsigned short getSubrateChannelA() const;
    void setSubrateChannelA(unsigned short x);
    UBYTE getNumBitsA() const;
    void setNumBitsA(UBYTE x);
    unsigned short getSubrateSpanB() const;
    void setSubrateSpanB(unsigned short x);
    unsigned short getSubrateChannelB() const;
    void setSubrateChannelB(unsigned short x);
    UBYTE getNumBitsB() const;
    void setNumBitsB(UBYTE x);
    UBYTE getAction() const;
    void setAction(UBYTE x);
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x000D)	3, 4	Message Type (0x000D)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB) See Response Status Values.
:	Address Method 0x00 - Individual AEs		
	Number of AEs to follow 0x02	10	Checksum
	AEs 0x25 - Subrate Channel		
:	Action 0x01 1-way Subrate Connection 0x02 2-way Subrate Connection 0x03 Disconnect		
:	Checksum		

Address Element 0x25 - Subrate Channel

Use the Subrate Channel Address Type to address two subrate channels.

For a 1-way connection, enter the source subrate channel in Address Element 1 and the destination subrate channel in Address Element 2.

For a 1-way disconnect, enter the subrate channel to be disconnected in both Address Element 1 and Address Element 2.

Byte	Description	Value
Address Element 1: Subrate Channel A		
2	Address Type	0x25
3	Data Length	0x05
4	Data[0] - Logical Span ID, MSB	
5	Data[1] - Logical Span ID, LSB	
6	Data[2] - Subrate Channel ID, MSB	
7	Data[3] - Subrate Channel ID, LSB	
8	Data[4] - Number of Bits in Subrate Channel	
Address Element 2: Subrate Channel B		
9	Address Type	0x25
10	Data Length	0x05
11	Data[0] - Logical Span ID, MSB	
12	Data[1] - Logical Span ID, LSB	
13	Data[2] - Subrate Channel ID, MSB	
14	Data[3] - Subrate Channel ID, LSB	
15	Data[4] - Number of Bits in Subrate Channel	

SwitchBackFromStandby

Type	SwitchKit API message
Description	Use the <i>SK_SwitchBackFromStandby</i> message to initiate a switchback from a standby line card to the primary one. This message can only be sent if the action field of the message <i>SK_SwitchBackFromStandbyConfig</i> is set to 1 during configuration.
Sent by	Application or Converged Services Administrator (CSA).
Arguments	The following table shows the arguments that you can modify:

Argument	Description
Action	This field can be set to: <ul style="list-style-type: none"> 0 = graceful switchback after LLC determines that the standby card is not handling any calls. 1 = forced switchback immediately.
Node	Specifies the node where the switchback is supposed to happen.
StandbySlot	Specifies the standby slot of the redundant card.

C Structure

```
typedef struct {
    UBYTE StandbySlot;
    UBYTE Action;
} SK_SwitchBackFromStandby;
```

C Structure Response

```
typedef struct {
    int Status;
} SK_SwitchBackFromStandbyAck;
```

C++ Class

```
class SKC_SwitchBackFromStandby : public
    SKC_ToolkitMessage {
public:
    UBYTE getStandbySlot() const;
    void setStandbySlot(UBYTE x);
    UBYTE getAction() const;
    void setAction(UBYTE x);
};
```

C++ Class Response

```
class SKC_SwitchBackFromStandbyAck : public
    SKC_ToolkitAck {
public:
    int getStatus() const;
    void setStatus(int x);
```

SwitchBackFromStandby

};

SwitchMgrQuery

Type SwitchKit API message

Description Use the *SK_SwitchMgrQuery* to allow an application to find out the current configuration state of the SwitchManager. If the SwitchManager is in the process of configuring when an application connects to LLC, *SK_SwitchMgrQuery* will notify the application of this state.

If you have the application/LLC set to channel recovery method 2 or 3, where channels are taken out of service, it is the application's responsibility to take the channels back into service when the first application watching a specific channel group connects to LLC using *ForceGroupState*. Without *SK_SwitchMgrQuery*, when SwitchManager is being started while the application is also starting up, there may be a condition where the application will take channels inservice prior to the configuration being complete. This can cause critical configuration messages from SwitchManager to fail since the channel is already in-service.

Sent by Application or Converged Services Administrator (CSA).

Arguments The following table shows the arguments that you can modify:

Argument	Description
QueryType	This field can be set to: <ul style="list-style-type: none"> • 0 = SK_SMQ_CONFIG_STATUS • 0 = SK_SMQ_IDLE • 1 = SK_SMQ_CONFIGURING

C Structure

```
typedef struct {
    UBYTE QueryType;
    UBYTE reserved18[252];
} SK_SwitchMgrQuery;
```

C Structure Response

```
typedef struct {
    unsigned short DataSize;
    int Status;
    UBYTE TLVCount;
    UBYTE Data[246];
} SK_SwitchMgrQueryAck;
```

C++ Class `class SKC_SwitchMgrQuery : public SKC_ToolkitMessage {`

```
public:  
    UBYTE getQueryType() const;  
    void setQueryType(UBYTE x);  
};
```

C++ Class Response

```
class SKC_SwitchMgrQueryAck : public SKC_ToolkitAck {  
public:  
    unsigned short getDataSize() const;  
    void setDataSize(unsigned short x);  
    int getStatus() const  
    void setStatus(int x);  
    UBYTE getTLVCount() const;  
    void setTLVCount(UBYTE x);  
    const UBYTE *getData() const;  
    UBYTE *getData();  
    void setData(UBYTE *x);  
};
```

Synchronization Priority List Configure 0x0006

SwitchKit Name	SynchPriorityConfig
Type	EXS API and SwitchKit API message
Description	<p>Synchronization Priority List Configure 0x0006</p> <p>Use this message to configure the synchronization priority list. At least one of the sources must be set to Free Running.</p> <p>The default priority for synchronization resources is as follows:</p> <ol style="list-style-type: none"> 1. External Reference Clock 1 (Primary) 2. External Reference Clock 2 (Secondary) 3. Loop Timing 1 (Primary) 4. Loop Timing 2 (Secondary) 5. Free Running clock (Internal)
Sent by	Host

SwitchKit Code Configuration

```
SynchPriorityConfig (
    Node = integer,
    Priority1Mode = integer,
    Priority2Mode = integer,
    Priority3Mode = integer,
    Priority4Mode = integer,
    Priority5Mode = integer);
```

C Structure

```
typedef struct {
    UBYTE Priority1Mode;
    UBYTE Priority2Mode;
    UBYTE Priority3Mode;
    UBYTE Priority4Mode;
    UBYTE Priority5Mode;
} XL_SynchPriorityConfig;
```

C++ Class

```
class XLC_SynchPriorityConfig : public
    XLC_OutboundMessage {
public:
    UBYTE getPriority1Mode() const;
    void setPriority1Mode(UBYTE x);
    UBYTE getPriority2Mode() const;
    void setPriority2Mode(UBYTE x);
    UBYTE getPriority3Mode() const;
    void setPriority3Mode(UBYTE x);
    UBYTE getPriority4Mode() const;
    void setPriority4Mode(UBYTE x);
    UBYTE getPriority5Mode() const;
```

Synchronization Priority List Configure 0x0006

```
void setPriority5Mode(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x000A)	1, 2	Length (0x0007)
3, 4	Message Type (0x0006)	3, 4	Message Type (0x0006)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status (MSB, LSB)
		10	Checksum
8	Priority 1 Synchronization Mode 0x01 Primary Reference (default) 0x02 Secondary Reference 0x03 Primary Loop 0x04 Secondary Loop 0x05 Free Running		
9	Priority 2 Synchronization Mode 0x01 Primary Reference 0x02 Secondary Reference (default) 0x03 Primary Loop 0x04 Secondary Loop 0x05 Free Running		
10	Priority 3 Synchronization Mode 0x01 Primary Reference 0x02 Secondary Reference 0x03 Primary Loop (default) 0x04 Secondary Loop 0x05 Free Running		
11	Priority 4 Synchronization Mode 0x01 Primary Reference 0x02 Secondary Reference 0x03 Primary Loop 0x04 Secondary Loop (default) 0x05 Free Running		
12	Priority 5 Synchronization Mode 0x01 Primary Reference 0x02 Secondary Reference 0x03 Primary Loop 0x04 Secondary Loop 0x05 Free Running (default)		
13	Checksum		

Synchronization Priority List Query 0x0081

SwitchKit Name	SynchPriorityQuery
Type	EXS API and SwitchKit API message
Description	Synchronization Priority List Query 0x0081 This message is used to query the system's clock source priority list, and the source currently in use.
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {
    } XL_SynchPriorityQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE CurrentMode;
    UBYTE Priority1Mode;
    UBYTE Priority2Mode;
    UBYTE Priority3Mode;
    UBYTE Priority4Mode;
    UBYTE Priority5Mode;
    } XL_SynchPriorityQueryAck;
```

C++ Class

```
class XLC_SynchPriorityQuery : public XLC_OutboundMessage
{
public:
    };
```

C++ Class Response

```
class XLC_SynchPriorityQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getCurrentMode() const;
    void setCurrentMode(UBYTE x);
    UBYTE getPriority1Mode() const;
    void setPriority1Mode(UBYTE x);
    UBYTE getPriority2Mode() const;
    void setPriority2Mode(UBYTE x);
    UBYTE getPriority3Mode() const;
    void setPriority3Mode(UBYTE x);
    UBYTE getPriority4Mode() const;
    void setPriority4Mode(UBYTE x);
    UBYTE getPriority5Mode() const;
    void setPriority5Mode(UBYTE x);
```

};

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x0005)	1, 2	Length (0x000D)
3, 4	Message Type (0x0081)	3, 4	Message Type (0x0081)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Checksum	8, 9	Status MSB, LSB
Response continued below.			
10	Current Synchronization Mode 0x01 Primary Reference Clock Signal 0x02 Secondary Reference Clock Signal 0x03 Primary Loop Clock Signal 0x04 Secondary Loop Clock Signal 0x05 Free Running Clock Signal		
11	Priority 1 Synchronization Mode 0x01 Primary Reference Clock Signal 0x02 Secondary Reference Clock Signal 0x03 Primary Loop Clock Signal 0x04 Secondary Loop Clock Signal 0x05 Free Running Clock Signal		
12	Priority 2 Synchronization Mode 0x01 Primary Reference Clock Signal 0x02 Secondary Reference Clock Signal 0x03 Primary Loop Clock Signal 0x04 Secondary Loop Clock Signal 0x05 Free Running Clock Signal		
13	Priority 3 Synchronization Mode 0x01 Primary Reference Clock Signal 0x02 Secondary Reference Clock Signal 0x03 Primary Loop Clock Signal 0x04 Secondary Loop Clock Signal 0x05 Free Running Clock Signal		

Synchronization Priority List Query 0x0081

14	Priority 4 Synchronization Mode 0x01 Primary Reference Clock Signal 0x02 Secondary Reference Clock Signal 0x03 Primary Loop Clock Signal 0x04 Secondary Loop Clock Signal 0x05 Free Running Clock Signal
15	Priority 5 Synchronization Mode 0x01 Primary Reference Clock Signal 0x02 Secondary Reference Clock Signal 0x03 Primary Loop Clock Signal 0x04 Secondary Loop Clock Signal 0x05 Free Running Clock Signal
16	Checksum

System Configuration 0x00AF

SwitchKit Name SystemConfig

Type EXS API and SwitchKit API message

Description **System Configuration 0x00AF**

This message configures system, non-related options.

Those options include:

- Message Resend Logic
- Host Link Failure Detection Logic
- System Busy Warning
- Resource Threshold

Sent by Host Application

Example Message (Socket Log Output for SwitchKit)

The following socket log output/example message shows the line card in slot 01 configured not to resend unacknowledged messages.

```
00 07 00 AF 00 00 FF 02 00
```

SwitchKit Code **Configuration**

```
SystemConfig (
    Node = integer,
    ConfigType = integer,
    Data = byte array);
```

NOTE: Data must contain the Slot AIB first:

```
0x00 0x01 0x01 0x01 SlotNumber
then the Data of the message as usual.
```

Example

This is an example of a SwitchManager message:

```
SystemConfig(Node=1, ConfigType=9,
    Data=0x00:0x01:0x01:0x01:0x05:
    0x01:0x04:0x00:0x01:0x02:0x03);
```

In this example, the slot number is: 0x00:0x01:0x01:0x01 in the slot AIB; and the data parameters of the message are: 0x01:0x04:0x00:0x01:0x02:0x03.

C Structure

```
typedef struct {
    UBYTE ConfigType;
```

```

    UBYTE Data[252];
} XL_SystemConfig;

```

C++ Class

```

class XLC_SystemConfig : public XLC_OutboundMessage {
public:
    UBYTE getConfigType() const;
    void setConfigType(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};

```

Overview of message The following table provides an overview of this message. The table following it provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00AF)	3, 4	Message Type (0x00AF)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Configuration Type	8, 9	Status MSB, LSB
9	<u>AIB</u>	10	Checksum
:	Address Method		
	Number of AEs to follow		
	AEs		
:	Data		
:	:		
:	Checksum		

**EXS API Hex Format-
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00AF)	3, 4	Message Type (0x00AF)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Configuration Type	8, 9	Status MSB, LSB
		10	Checksum
	0x00 Reserved 0x01 Reserved 0x02 Message Resend Logic 0x03 Host Link Failure Detection Logic 0x06 System Busy Warning 0x07 Resource Threshold 0x09 Resource Usage Reporting 0x0B Approaching Busy Threshold for Slots 0x0C Real Busy Threshold for Slots 0x0D Message Grouping 0x11 DSP Resource Threshold Alarm 0x17 Extended Conference Enable		
9	<u>AIB</u> : Address Method Used by the following Configuration Types only: 0x03 Host Link Failure Detection Logic 0x07 Resource Threshold 0x09 Resource Usage Reporting 0x0B Approaching Busy Threshold for Slots 0x0C Real Busy Threshold for Slots		
	Number of AEs		
	AEs 0x01 Slot Must be included if Host Link Failure is for line cards.		
:	Data[0] (Data depends on the Configuration Type)		

: **0x02 Message Resend Logic**

Data[0] Options

- 0x00 Do Not Resend
Does not resend unacknowledged messages.
- 0x01 Resend Once (Default)*
- 0x02 Resend Once and Generate *Alarm* Message
- 0x03 Resend Until Acknowledged*
Resends the message until acknowledged by the host.
This applies to the following messages only: *Alarm*, *Alarm Cleared*,
Request For Service, *Request For Service with Data*, and
Channel Released.
- 0x04 Disable "Resend Until Acknowledged"
Do not resend the messages listed above until acknowledged by the
host. This cancels "Resend Until Acknowledged."

*To enable "Resend Until Acknowledged" you must also enable "Resend Once"

0x03 Host Link Failure Detection Logic

Data[0] Feature Enable/Disable

- 0x00 Disable Host Link Failure Detection (Default)
- 0x01 Enable Host Link Failure Detection

Data[1] Response to Host Link Failure (HLF)

Bit 0 Setting for whether spans and channels will be taken out-of-service
0 Set all channels out of service when host link failure detected (Default)
1 Set all spans and channels out of service when host link failure
detected

Bit 1 Setting for whether a response will be sent when an HLF is detected
0 A response will be sent
1 No response will be sent

Bit 2 Setting for taking SCCP/TCAP out-of-service
0 Do not take SCCP/TCAP out-of-service
1 Take SCCP/TCAP out-of-service

Data[2,3] Failure Detection Timer
Value expressed in 10 ms units

NOTE: The span/channels continue in the out-of-service state until the
host manually brings them back in-service.

0x06 System Busy Warning

Data[0] Sensitivity Level
0x00 Least Sensitive
0x01 :
0x02 :
0x03 :
0x04 :
0x05 Most Sensitive

Data[1] System Busy Warning

0x00 Disable (Default)
0x01 Enable

0x07 Resource Threshold

Data[0] Resource Type
0x01 Conferencing Timeslots

Data[1,2] Number of Assigned Timeslots to Generate an *Alarm Cleared* message
Data[3,4] Number of Assigned Timeslots to Generate an *Alarm* message

0x09 Resource Usage Reporting

Data[0] 0x00 Disable (Default)
0x01 Enable (Default interval is 5 minutes)
0x02 Change the Reporting Interval

Data[1] If Data[0] above is 0x00 (Disable) or 0x01 (Enable):
Number of Resources

If Data[0] above is 0x02 (Change the Reporting Interval):
Number of Minutes (Max: 254)

Data [2-N] Resource ID 1- Resource ID N:

NOTE: If Data[0] above is 0x02, Data[2-N] do not apply.

0x00 Memory
0x01 MCB
0x02 CPU Level 1
0x03 CPU Level 2

0x0B Approaching Busy Threshold for Slots

Data[0]

0x00 Disable

0x01 Enable

Default value are as follows:

Memory - 70 percent

MCB -70 percent

CPU - 80 percent.

0x02 Change Thresholds

Data[1] Number of Resources

Data[2-N] Resource ID 1- Resource ID N:

0x00 Memory (percentage, expressed in Hex)

0x01 MCB (percentage, expressed in Hex)

0x02 CPU Usage (percentage, expressed in Hex)

0x0C Real Busy Threshold for Slots

Data[0]

0x00 Disable Alarm (Default)

0x01 Enable Alarm

Default value are as follows:

Memory - 84 percent

MCB -84 percent

CPU - 90 percent.

Data[1] Number of Resources

Data[2-N] Resource ID 1- Resource ID N:

0x00 Memory (percentage, expressed in Hex)

0x01 MCB (percentage, expressed in Hex)

0x02 CPU Usage (percentage, expressed in Hex)

0x0D Message Grouping (Supported on ONE-series cards only)

Data[0]

- 0x00 Disable (Default)
- 0x01 Enable

0x11 DSP Resource Threshold Alarm

Data[0] DSP Function Type

- 0xFF Configure **all** DSP Function Types (except conferences) with the percentage values in Data[1] and Data[2] below

- 0x00 No Function Type

Tone Reception

- 0x01 DTMF μ -law (all DSP cards)
- 0x02 MFR1 μ -law (all DSP cards)
- 0x03 DTMF A-Law (all DSP cards)
- 0x04 MFR1 A-Law (all DSP cards)
- 0x05 MFR2 A-Law (all DSP cards)
- 0x06 MFR2 μ -Law (all DSP cards)
- 0x07 CPA A-Law (all DSP cards)
- 0x08 CPA μ -Law (all DSP cards)
- 0x09 E1 Dial Pulse (MFDSP and DSP-ONE cards only)
- 0x0A Energy Detection (all DSP cards)
- 0x0C DTMF High Pass Filter μ -Law (MFDSP card only)
- 0x0D DTMF High Pass Filter A-Law (MFDSP card only)
- 0x0E Coin Detection μ -Law (DSP-ONE card only)
- 0x0F Coin Detection A-Law (DSP-ONE card only)

Tone Generation

- 0x10 DTMF μ -law
- 0x11 DTMF A-law
- 0x12 MFR1 μ -law
- 0x13 MFR1 A-law
- 0x14 CPT4 μ -law
- 0x15 CPT4 A-law
- 0x16 MFR2 Forward A-law
- 0x17 MFR2 Forward μ -law
- 0x18 MFR2 Backward A-law
- 0x19 MFR2 Backward μ -law
- 0x1A Bong Tone μ -law
- 0x1B Bong Tone A-law
- 0x30 Universal Tone Generation μ -law
- 0x31 Universal Tone Generation A-law

	<p><u>Conferencing</u></p> <p>0x1E Standard μ-Law (MFDSP and DSP-ONE cards only)</p> <p>0x1F Standard A-Law (MFDSP and DSP-ONE cards only)</p> <p>0x20 Mixed (MFDSP and DSP-ONE cards only)</p> <p>0x21 Monitor (all DSP cards)</p> <p>0x22 Unified (DSP-ONE and DSP 2 cards only)</p> <p>0x23 DTMF-Clamped Conference (DSP-ONE and DSP 2 cards only)</p> <p><u>Announcement Generation</u></p> <p>0x1C VRAS (MFDSP and DSP-ONE cards only)</p> <p><u>File Playback / Record</u></p> <p>0x1D File Playback / Record (DSP 2 card only)</p> <p>Data[1] 0-100 Percentage of resources to exceed to produce alarm (default =100).</p> <p>Data[2] 0-100 Percentage of resources to go below to clear alarm (default =0).</p> <p>0x17 Extended Conference Enable</p> <p>Data[0]</p> <p>0x00 Disable Extended Conferences</p> <p>0x01 Enable Extended Conferences</p>
:	Checksum

System Configuration Query 0x00B4

SwitchKit Name SystemConfigQuery

Type EXS API and SwitchKit API message

Description **System Configuration Query 0x00B4**

Use this message to query various configuration settings in the CSP. Some query results are reported in the response to the message, while others generate a *Generic Report* message.

NOTE: The acknowledgment for Active Conference IDs (query type 0x04) will return partial data — only the number of conferences. The report will have the remaining data, that is, all the conference IDs.

The data for the following Query Types is returned in the response:

- 0x02 - Message Resend Logic
- 0x03 - Host Link Failure Detection Logic
- 0x04 - Active Conference IDs
- 0x06 - System Busy Warning
- 0x17 - Extended Conference Enable

The data for the following Query Types is returned in the *Generic Report* message:

- 0x04 - Active Conference IDs
- 0x05 - Channel State
- 0x07 - Resource Threshold
- 0x09 - Resource Usage Reporting
- 0x0B - Approaching Busy Threshold for Slots
- 0x0C - Real Busy Threshold for Slots
- 0x10 - Detailed Conference Information
- 0x11 - DSP Resource Threshold
- 0x13 - Conference Speakers
- 0x14 - Conferencing Features
- 0x15 - Child Conference Information
- 0x16 - Child Conference IDs
- 0x18 - Cache File Query

For SwitchKit

To get this report, you must register for it by calling the `sk_msgRegister` function with the parameter, `SK_RESOURCE_UTIL_REPORT`.

Sent by Host Application

SwitchKit Code C Structure

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE QueryType;
    UBYTE Data;
    UBYTE reserved49[221];
} XL_SystemConfigQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE reserved6[13];
    UBYTE Data[251];
} XL_SystemConfigQueryAck;
```

C++ Class

NOTE: The functions, `getDataBlock()` and `setDataBlock()`, should be used only for `QueryType 0x18`. The `setData()` function should not be used for this `QueryType`.

```
class XLC_SystemConfigQuery : public XLC_OutboundMessage
{
    public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    XBYTE getStartSpan() const;
    void setStartSpan(XBYTE x);
    UBYTE getStartChannel() const;
    void setStartChannel(UBYTE x);
    XBYTE getEndSpan() const;
    void setEndSpan(XBYTE x);
    UBYTE getEndChannel() const;
    void setEndChannel(UBYTE x);
    XBYTE getConferenceID() const;
    void setConferenceID(XBYTE x);
    XBYTE getParentConferenceID() const;
    void setParentConferenceID(XBYTE x);
    XBYTE getChildConferenceID() const;
    void setChildConferenceID(XBYTE x);
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getQueryType() const;
    void setQueryType(UBYTE x);
    UBYTE *getData() const ;
    void setData(UBYTE *x) ;
    const UBYTE *getDataBlock() const;
    UBYTE *getDataBlock();
    void setDataBlock(UBYTE *x);
};
```

C++ Class Response class **XLC_SystemConfigQueryAck** : public
 XLC_AcknowledgeMessage {
 public:
 unsigned short getStatus() const;
 void setStatus(unsigned short x);
 const UBYTE *getData() const;
 UBYTE *getData();
 void setData(UBYTE *x);
 };

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00B4)	3, 4	Message Type (0x00B4)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status MSB, LSB

System Configuration Query 0x00B4

8	<p>AIB 0x00 - Individual AEs 0x01 - Range</p> <p>Depending on Query Type</p>	<p>Data</p> <p>The values in the Response's Data fields depend on the value of the Query Type field.</p> <p>There is no response data for the following Query Types. The data for these queries is returned in the <i>Generic Report</i> message.</p> <ul style="list-style-type: none"> 0x05 - Channel State 0x07 - Resource Threshold 0x09 - Resource Usage Reporting 0x0B - Approaching Busy Threshold for Slots 0x0C - Real Busy Threshold for Slots 0x10 - Detailed Conference Information 0x11 - DSP Resource Threshold 0x13 - Conference Speakers 0x14 - Conferencing Features 0x15 - Child Conference Information 0x16 - Child Conference IDs (data also returned in this message) 0x17 - Extended Conference Enable 0x18 - Cache File Query
9	<p>Number of AEs to follow</p>	
10	<p>AEs</p> <p>The Address Element used depends on the Query Type, as follows:</p> <ul style="list-style-type: none"> 0x02 Message Resend Logic 0x00 00 00 00 Null 0x03 Host Link Failure Detection Logic 0x00 00 00 00 Null 0x04 Active Conference IDs 0x00 00 00 00 Null 0x05 Channel State 0x0D Channel (A) 0x0D Channel (B) (For one channel enter same channel in both AEs). 0x06 System Busy Warning 0x00 00 00 00 Null 0x07 Resource Threshold 0x00 00 00 00 Null 0x09 Resource Usage Reporting 0x00 00 00 00 Null 0x0B Approaching Busy Threshold for Slots 0x01 Slot 0x0C Real Busy Threshold for Slots 0x01 Slot 0x10 Detailed Conference Information 0x55 Conference ID 0x11 DSP Resource Threshold 0x01 Slot 0x13 Conference Speakers 0x55 Conference ID 0x14 Conferencing Features 0x0D Channel (2 AEs with same Channel) 0x55 Conference ID or 0x45 Child Conference ID 0x15 Child Conference Information 0x45 Child Conference ID 0x16 Child Conference IDs 0x55 Conference ID (Parent) 0x18 Cache File Query 0x01 Slot 	<ul style="list-style-type: none"> 0x02 Message Resend Logic <ul style="list-style-type: none"> Data[0] Resend once <ul style="list-style-type: none"> 0x00 Disabled 0x01 Enabled 0x02 Enable and Generate <i>Alarm</i> message Data[1] Resend Until Acknowledged <ul style="list-style-type: none"> 0x03 Resend <i>Alarm</i>, <i>Alarm Cleared</i>, <i>Request for Service</i>, <i>Request for Service with Data</i>, and <i>Channel Release</i> until acknowledged by the host. 0x04 Do not resend messages in option 0x03 until acknowledged by the host (this option cancels option 0x03)

System Configuration Query 0x00B4

<p>: Query Type : 0x00 Reserved 0x01 Reserved 0x02 Message Resend Logic 0x03 Host Link Failure Detection Logic 0x04 Active Conference IDs 0x05 Channel State 0x06 System Busy Warning 0x07 Resource Threshold 0x09 Resource Usage Reporting 0x0B Approaching Busy Threshold for Slots 0x0C Real Busy Threshold for Slots 0x10 Detailed Conference Information 0x11 DSP Resource Threshold 0x13 Conference Speakers 0x14 Conferencing Features 0x15 Child Conference Information 0x16 Child Conference IDs 0x17 Extended Conference Enable 0x18 Cache File Query</p>	<p>0x03 Host Link Failure Detection Logic Data[0] Feature Enabled/Disabled 0x00 Disable 0x01 Enable Data[1] Response to Host Link Failure Bit 0 Setting of spans and channels 0 All channels out of service when host link failure detected (Default) * 1 All spans and channels out of service when host link failure detected Bit 1 Setting of response 0 A response is sent 1 No response is not sent Bit 2 Setting of SCCP/TCAP OOS 0 SCCP/TCAP is not taken out-of-service 1 SCCP/TCAP is taken out-of-service Data[2, 3] Failure Detection Timer 10 ms units (cannot be less than 30,000 ms.)</p>
--	--

<p>Data</p> <p>Resource Threshold (0x07) Resource Type</p> <p>DSP Resource Threshold (0x011) Function Type</p> <p><u>Tone Reception</u></p> <p>0x01 DTMF μ-law (all DSP cards) 0x02 MFR1 μ-law (all DSP cards) 0x03 DTMF A-Law (all DSP cards) 0x04 MFR1 A-Law (all DSP cards) 0x05 MFR2 A-Law (all DSP cards) 0x06 MFR2 μ-Law (all DSP cards) 0x07 CPA A-Law (all DSP cards) 0x08 CPA μ-Law (all DSP cards) 0x09 E1 Dial Pulse (MFDSP and DSP-ONE cards only) 0x0A Energy Detection (all DSP cards) 0x0C DTMF High Pass Filter μ-Law (MFDSP card only) 0x0D DTMF High Pass Filter A-Law (MFDSP card only) 0x0E Coin Detection μ-Law (DSP-ONE card only) 0x0F Coin Detection A-Law (DSP-ONE card only)</p> <p><u>Tone Generation</u></p> <p>0x10 DTMF μ-law 0x11 DTMF A-law 0x12 MFR1 μ-law 0x13 MFR1 A-law 0x14 CPT4 μ-law 0x15 CPT4 A-law 0x16 MFR2 Forward A-law 0x17 MFR2 Forward μ-law 0x18 MFR2 Backward A-law 0x19 MFR2 Backward μ-law 0x1A Bong Tone μ-law 0x1B Bong Tone A-law 0x30 Universal Tone Generation μ-law 0x31 Universal Tone Generation A-law</p>		
--	--	--

System Configuration Query 0x00B4

<p><u>Conferencing</u> 0x1E Standard μ-Law (MFDSP and DSP-ONE cards only) 0x1F Standard A-Law (MFDSP and DSP-ONE cards only) 0x20 Mixed (MFDSP and DSP-ONE cards only) 0x21 Monitor (all DSP cards) 0x22 Unified (DSP-ONE and DSP 2 cards only) 0x23 DTMF-Clamped Conference (DSP-ONE and DSP 2 cards only)</p> <p><u>Announcement Generation</u> 0x1C VRAS (MFDSP and DSP-ONE cards only)</p> <p><u>File Playback / Record</u> 0x1D File Playback / Record (DSP 2 card only)</p> <p>Data[1] 0-100 Percentage of resources to exceed to produce alarm (default =100). Data[2] 0-100 Percentage of resources to go below to clear alarm (default =0).</p> <p>Cache File Query (0x018) [0-3] File ID (applies only if Query Type is Cache File Query (0x18))</p> <p>For all other Query Types, the Data field does not apply.</p>	
--	--

System Log Query 0x0082

SwitchKit Name	SystemLogQuery
Type	EXS API and SwitchKit API message
Description	<p>System Log Query 0x0082</p> <p>This message is used by the host to inquire about counters the system uses to keep track of certain events.</p>
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {
    } XL_SystemLogQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    unsigned short TrunkStatePurges;
    unsigned short UnackedHostMsgs;
    unsigned short CallsNotServiced;
    unsigned short MsgRetriesControl;
    unsigned short MsgRetriesMatrix;
    } XL_SystemLogQueryAck;
```

C++ Class

```
class XLC_SystemLogQuery : public XLC_OutboundMessage {
public:
    };
```

C++ Class Response

```
class XLC_SystemLogQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    unsigned short getTrunkStatePurges() const;
    void setTrunkStatePurges(unsigned short x);
    unsigned short getUnackedHostMsgs() const;
    void setUnackedHostMsgs(unsigned short x);
    unsigned short getCallsNotServiced() const;
    void setCallsNotServiced(unsigned short x);
    unsigned short getMsgRetriesControl() const;
    void setMsgRetriesControl(unsigned short x);
    unsigned short getMsgRetriesMatrix() const;
    void setMsgRetriesMatrix(unsigned short x);
    };
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x0005)	1, 2	Length (0x0011)
3, 4	Message Type (0x0082)	3, 4	Message Type (0x0082)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Checksum	8, 9	Status MSB, LSB
Response continued below.			
10, 11	Number of Trunk Stage Purges (MSB, LSB)		
12, 13	Number of unacknowledged host messages (MSB, LSB)		
14, 15	Number of calls never serviced by host (MSB, LSB)		
16, 17	Number of message retries on control/status HDLC link (MSB, LSB)		
18, 19	Number of message retries on CSP Matrix Series 3 Card-to-CSP Matrix Series 3 Card HDLC link (MSB, LSB)		
20	Checksum		

NOTE: All counts are totals since last system power-up. All counts are reset when this message is received by the CSP.

System Resource Usage Query 0x008E

SwitchKit Name SystemResourceUtilQuery

Type EXS API and SwitchKit API message

Description **System Resource Usage Query 0x008E**

Use this message to query the utilization status of the following cards:
T-ONE, E-ONE, SS7 Series 3, SS7 PQ, DSP, and CSP Matrix Series 3 Card.

Sent by Host

Sample Message In the following sample, the host sends the System Resource Usage Query to the CSP to query the Resource Usage for a Card. Following is the response from the CSP.

Host-to-CSP

```
00 10 00 8e 00 00 ff 02 00 01 01 01 09 04 00 01 02 03
```

CSP-to-Host Response

```
00 5f 00 8e 00 00 ff 00 10 02 00 01 01 01 09 04 00 26 03 01  
02 00 02 48 c0 00 00 01 c0 03 00 02 63 00 00 00 00 00  
05 00 05 9e 00 00 03 28 00 05 00 08 8c 00 00 00 06 00  
01 01 00 02 08 00 00 01 a0 01 00 2d 19 03 1a 00 00 01  
a0 07 00 2d 19 02 00 32 04 00 4b 05 00 4b 06 01 00 07  
33 4b 32 03 00
```

SwitchKit Code **C Structure**

```
typedef struct {  
    UBYTE QueryType;  
    UBYTE Slot;  
    UBYTE NumOfResources;  
    UBYTE Data[221];  
} XL_SystemResourceUtilQuery;
```

C Structure Response

```
typedef struct {  
    unsigned short Status;  
    UBYTE QueryType;  
    UBYTE Data[250];  
} XL_SystemResourceUtilQueryAck;
```

C++ Class

```
class XLC_SystemResourceUtilQuery : public
    XLC_OutboundMessage {
public:
    UBYTE getQueryType() const;
    void setQueryType(UBYTE x);
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    UBYTE getNumOfResources() const;
    void setNumOfResources(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

C++ Class Response

```
class XLC_SystemResourceUtilQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getQueryType() const;
    void setQueryType(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x008E)	3, 4	Message Type (0x008E)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Query Type 0x01 Conferencing Resources 0x02 Resource Usage for a Card	8, 9	Status MSB, LSB
NOTE: The following fields apply only if the Query Type field is 0x02			
9	<u>AIB</u> : Address Method 0x00 - Individual AEs		
	Number of AEs to follow		
	AEs 0x01 Slot		
:	Number of Resources		
:	Resource ID 1 0x00 Memory 0x01 MCB 0x02 CPU Usage Level 1 0x03 CPU Usage Level 2		
:	Checksum	10	Query Type 0x01 Conferencing Resources 0x02 Resource Usage for a Card
			Note: The following fields apply only if the Query Type field is 0x02.
			If the Query Type is 0x01 proceed to the Data below.
		:	AIB Address Method 0x00 - Individual AEs
		:	Number of AEs to follow.
		:	AEs 0x01 Slot
		:	Number of Resources
		:	Data (continued below)

(For Query Type 0x01)

Data[0,1] Timeslots in use (MSB,LSB)
 Data[2,3] Timeslots available (MSB,LSB)
 Data[4,5] Percent used (0-100)

(For Query Type 0x02)

0x00 Resource ID: Memory

Data[0] Length
 Data[1] Number of regions configured on the card
 Data[2] Number of partitions configured on the card
 Data[3] Region ID of Nth region
 Data[4-7] Size of the region in bytes
 Data[8-11] Number of bytes used from the region
 Data [12] Partition ID of Nth partition
 Data[13-16] Size of the partition in bytes
 Data[17-20] Used bytes of partition
 :
 Data[:] Region ID of Nth region(1 byte)
 Data[:] Size of the Nth region in bytes (4 bytes)
 Data[:] Number of bytes used from the Nth region(4 bytes)
 Data [:] Partition ID of 1st partition (1 byte)
 Data[:] Size of the 1st partition in bytes (4 bytes)
 Data[:] Used bytes of 1st partition (4 bytes)
 :
 Data [:] Partition ID of Nth partition (1 byte)
 Data[:] Size of the Nth partition in bytes (4 bytes)
 Data[:] Used bytes of Nth partition (4 bytes)

Repeat for each region

Data[n] Region ID of Nth region
 Data[n+1 - n+4] Size of the region in bytes
 Data[n+5 - n+8] Number of bytes used from the region

Repeat for each segment

Data[n] Partition ID of Nth partition
 Data[n+1 - n+4] Size of the partition in bytes
 Data[n+5 - n+8] Number of bytes used from the partition

0x01 Resource ID: MCB

Data[0] Length
 Data[1] Message buffer usage

0x02 Resource ID: CPU Usage Level 1

Data[0] Length
 Data[1-4] Time in milliseconds covering this report
 Data[5] Number of tasks reported (0x01)
 Data[6] Logical task ID (idle task only 0x00)
 Data[7] Percentage of time usage for task (integer part)
 Data[8] Percentage of time usage for task (decimal part)

0x03 Resource ID: CPU Usage Level 2

Data[0] Length
 Data[1-4] Time in milliseconds covering this report
 Data[5] Number of tasks reported (the following repeats for each task)
 Data[6] Logical task ID
 Data[7] Percentage of time usage for task (integer part)
 Data[8] Percentage of time usage for task (decimal part)

0x04 Resource ID: CPU Usage Level 3

System Resource Usage Query 0x008E

15	Checksum
----	----------

T1 Span Configure 0x00A9

SwitchKit Name T1SpanConfig
Type EXS API and SwitchKit API message

Description **T1 Span Configure 0x00A9**
This message is used to configure the characteristics of a T1 span.

NOTE: Before changing span configuration, Dialogic ALWAYS RECOMMENDS that you de-assign the spans, assign them again and then send the new span configuration.

Sent by Host

Example Message (Socket Log Output for SwitchKit) The following socket log output/example message shows a T1 Span being configured with D4 framing and a line coding method of bit 7 zero suppression. The line length is 0-133 feet.

```
00 0D 00 A9 00 00 FF 00 01 0C 02 01 23 09 06
```

SwitchKit Code **Configuration**

```
T1SpanConfig (  
    Node = integer,  
    Span = integer,  
    Format1 = integer,  
    Format2 = integer);
```

NOTE: In SwitchKit, the parameter Format1 is the same as Format in EXS API. In SwitchKit, the parameter Format2 is the same as Line Length in EXS API.

C Structure

```
typedef struct {  
    unsigned short Span;  
    UBYTE Format1;  
    UBYTE Format2;  
} XL_T1SpanConfig;
```

C++ Class

```
class XLC_T1SpanConfig : public XLC_SpanMessage {  
public:  
    unsigned short getSpan() const;  
    void setSpan(unsigned short x);  
    UBYTE getFormat1() const;  
    void setFormat1(UBYTE x);  
    UBYTE getFormat2() const;
```

```
void setFormat2(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)																			
Byte	Field Description	Byte	Field Description																		
0	Frame (0xFE)	0	Frame (0xFE)																		
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)																		
3, 4	Message Type (0x00A9)	3, 4	Message Type (0x00A9)																		
5	Reserved (0x00)	5	Reserved (0x00)																		
6	Sequence Number	6	Same Sequence Number																		
7	Logical Node ID	7	Logical Node ID																		
8	<u>AIB</u>	8, 9	Status MSB, LSB																		
:	Address Method	10	Checksum																		
:	0x00 - Individual AEs (NOT Ranges)																				
	Number of AEs to follow																				
	AEs																				
	0x0C Logical Span																				
:	<p>Format</p> <p>This field is a bit mask. The bit values are 0=Disabled, 1=Enabled.</p> <table border="0"> <tr> <td><u>Bit</u></td> <td><u>Framing</u></td> </tr> <tr> <td>0</td> <td>D4 (Default)</td> </tr> <tr> <td>1</td> <td>ESF</td> </tr> <tr> <td>2</td> <td>Reserved, must be 0</td> </tr> </table> <p>Line Coding Method</p> <table border="0"> <tr> <td>3</td> <td>Bit 7 zero suppressing (Default)</td> </tr> <tr> <td>4</td> <td>B8ZS zero suppression</td> </tr> <tr> <td>5</td> <td>Reserved, must be 0</td> </tr> </table> <p>Signaling Method</p> <table border="0"> <tr> <td>6</td> <td>Clear Channel (disable signaling insertion)</td> </tr> <tr> <td>7</td> <td>Reserved, must be 0</td> </tr> </table> <p>NOTE: To configure a single node for greater than 64 spans, please contact Dialogic technical support.</p>			<u>Bit</u>	<u>Framing</u>	0	D4 (Default)	1	ESF	2	Reserved, must be 0	3	Bit 7 zero suppressing (Default)	4	B8ZS zero suppression	5	Reserved, must be 0	6	Clear Channel (disable signaling insertion)	7	Reserved, must be 0
<u>Bit</u>	<u>Framing</u>																				
0	D4 (Default)																				
1	ESF																				
2	Reserved, must be 0																				
3	Bit 7 zero suppressing (Default)																				
4	B8ZS zero suppression																				
5	Reserved, must be 0																				
6	Clear Channel (disable signaling insertion)																				
7	Reserved, must be 0																				
:	<p>Line Length</p> <table border="0"> <tr> <td>0x06</td> <td>000–133 ft. (Default)</td> </tr> <tr> <td>0x01</td> <td>134–166 ft.</td> </tr> <tr> <td>0x05</td> <td>167–299 ft.</td> </tr> <tr> <td>0x03</td> <td>300–533 ft.</td> </tr> <tr> <td>0x07</td> <td>534–655 ft.</td> </tr> <tr> <td>0x00</td> <td>G.703 ITU-T</td> </tr> </table>			0x06	000–133 ft. (Default)	0x01	134–166 ft.	0x05	167–299 ft.	0x03	300–533 ft.	0x07	534–655 ft.	0x00	G.703 ITU-T						
0x06	000–133 ft. (Default)																				
0x01	134–166 ft.																				
0x05	167–299 ft.																				
0x03	300–533 ft.																				
0x07	534–655 ft.																				
0x00	G.703 ITU-T																				
:	Checksum																				

T1 Span Query 0x0085

SwitchKit Name	T1SpanFormatQuery
Type	EXS API and SwitchKit API message
Description	T1 Span Query 0x0085 This message is used to query the configuration of a T1 span.
Sent by	Host
SwitchKit Code	C Structure

```
typedef struct {
    unsigned short Span;
} XL_T1SpanFormatQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    unsigned short Span;
    UBYTE Format;
    UBYTE LineLength;
} XL_T1SpanFormatQueryAck;
```

C++ Class

```
class XLC_T1SpanFormatQuery : public XLC_OutboundMessage
{
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
};
```

C++ Class Response

```
class XLC_T1SpanFormatQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    UBYTE getFormat() const;
    void setFormat(UBYTE x);
    UBYTE getLineLength() const;
    void setLineLength(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)																							
Byte	Field Description	Byte	Field Description																						
0	Frame (0xFE)	0	Frame (0xFE)																						
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)																						
3, 4	Message Type (0x0085)	3, 4	Message Type (0x0085)																						
5	Reserved (0x00)	5	Reserved (0x00)																						
6	Sequence Number	6	Same Sequence Number																						
7	Logical Node ID	7	Logical Node ID																						
8	AIB	8, 9	Status MSB, LSB																						
:	Address Method 0x00 - Individual AEs																								
	Number of AEs to follow	10	AIB (same as message)																						
	AE 0x0C Logical Span																								
:	Checksum																								
Response continued below.																									
:	Format This field is a bit mask. The meaning of each bit is listed below. The bit values are 0=Disabled, 1=Enabled. <table border="0"> <tr> <td><u>Bit</u></td> <td><u>Framing</u></td> </tr> <tr> <td>0</td> <td>D4</td> </tr> <tr> <td>1</td> <td>ESF</td> </tr> <tr> <td>2</td> <td>Reserved, must be 0</td> </tr> <tr> <td colspan="2">Line Coding Method</td> </tr> <tr> <td>3</td> <td>Bit 7 zero suppressing</td> </tr> <tr> <td>4</td> <td>B8ZS zero suppression</td> </tr> <tr> <td>5</td> <td>Reserved, must be 0</td> </tr> <tr> <td colspan="2">Signaling Method</td> </tr> <tr> <td>6</td> <td>Clear Channel (disable signaling insertion)</td> </tr> <tr> <td>7</td> <td>Reserved, must be 0</td> </tr> </table>			<u>Bit</u>	<u>Framing</u>	0	D4	1	ESF	2	Reserved, must be 0	Line Coding Method		3	Bit 7 zero suppressing	4	B8ZS zero suppression	5	Reserved, must be 0	Signaling Method		6	Clear Channel (disable signaling insertion)	7	Reserved, must be 0
<u>Bit</u>	<u>Framing</u>																								
0	D4																								
1	ESF																								
2	Reserved, must be 0																								
Line Coding Method																									
3	Bit 7 zero suppressing																								
4	B8ZS zero suppression																								
5	Reserved, must be 0																								
Signaling Method																									
6	Clear Channel (disable signaling insertion)																								
7	Reserved, must be 0																								
:	Line Length 0x06000–133 ft. 0x01134–166 ft. 0x05167–299 ft. 0x03300–533 ft. 0x07534–655 ft. 0x00G.703 ITU-T																								
	AIB 0x13 Physical Span																								
:	Checksum																								

Tag Configuration 0x00D0

SwitchKit Name TagConfig

Type EXS API and SwitchKit API message

Description **Tag Configuration 0x00D0**

Each line card and CSP Matrix Series 3 Card card has battery-backed RAM that stores configuration information provided by the host. This message allows the host to assign a tag to a card's configuration. after it has been configured. If the card configuration is modified, due to either CSP restoration of default values or host-initiated configuration commands, the tag is reset to 0.

The tag is reported to the host in the *Card Status Report* and *Card Status Query* messages. With this feature, the host can quickly determine if a card has the proper configuration. When a card's configuration is modified, the tag is reset to 0, and the card must be retagged. In the event that the system is powered down, or a card is inserted, all battery-backed configuration is cleared.

Sent by Host

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE Slot;
    unsigned short BrdTag;
} XL_TagConfig;
```

C++ Class

```
class XLC_TagConfig : public XLC_OutboundMessage {
public:
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    unsigned short getBrdTag() const;
    void setBrdTag(unsigned short x);
};
```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00D0)	3, 4	Message Type (0x00D0)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8, 9	Status MSB, LSB
:	Address Method		
	Number of AEs to follow	10	Checksum
	AEs		
	Tag MSB, LSB		
:	Checksum		

**EXS API Hex Format -
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x00D0)	3, 4	Message Type (0x00D0)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8, 9	Status MSB, LSB
:	Address Method	10	Checksum
	0x00 - Individual AEs		
	Number of AEs to follow		
	AEs		
	0x01 Slot		
:	Tag MSB, LSB A two-byte number assigned by the host to identify a card's configuration. NOTE: The Tag value 0x0000 is reserved for untagged configurations. The following is a list of the messages that would reset the configuration tag of a line card only if they are successfully processed (for example, the host receives a response Status of Positive Acknowledgment, 0x10).		

T-ONE, E-ONE, DS3

These messages reset the configuration tags of T-ONE, E-ONE, and DS3 cards:

- Answer Supervision Mode Configure
- Assign Logical Span ID
- B Channel Configure
- Busy Out Flag Configure
- D Channel Assign
- D Channel De-assign
- D Channel Facility List Configure
- DS3 Configure-Query (DS3 card only - when changing DS3 framer)
- Distant End Release Mode Configure
- E1 Span Configure
- Filter Timer Configure
- Flash Timing Configure
- Impulsing Parameter Configure
- Inseize Instruction List Configure
- Local End Release Mode Configure
- PCM Encoding Format Configure
- PPL Assign
- PPL Audit Configure
- PPL Configure
- PPL Timer Configure
- PPL Transmit Signal Configure
- Product License Download
- Receive Signaling Configure
- Service State Configure
- Span Filter Configure
- Start Dial Configure
- T1 Span Configure
- Transmit Signaling Configure
- Trunk Type Configure

J-ONE, SJ1LC

These messages reset the configuration tags of J-ONE cards:

- Assign Logical Span ID
- D Channel De-assign
- J1 Span Configure
- Service State Configure
- Span Filter Configure

DSP

These messages will reset the configuration tags DSP-ONE cards:

- Call Progress Analysis Class Configure
- Call Progress Analysis Pattern Configure
- DSP SIMM Configure
- Service State Configure
- Transmit Cadence Pattern Configure
- Transmit Tone Configure

Tag Configuration 0x00D0

	<p>ISDN PRI</p> <p>These messages will reset the configuration tags of ISDN PRI cards:</p> <ul style="list-style-type: none">Answer Supervision Mode ConfigureB Channel ConfigureD Channel AssignD Channel De-assignD Channel Facility List ConfigureDistant End Release Mode ConfigureISDN Interface ConfigureISDN Profile ConfigureISDN Terminal ConfigureLocal End Release Mode ConfigureOutseize Instruction List ConfigurePCM Encoding Format ConfigurePPL AssignPPL Audit ConfigurePPL ConfigurePPL DeletePPL Transmit Signal ConfigureService State Configure
	<p>SS7</p> <p>These messages will reset the configuration tags of SS7 cards:</p> <ul style="list-style-type: none">Answer Supervision Mode ConfigureDistant End Release Mode ConfigureLocal End Release Mode ConfigureOutseize Instruction List ConfigurePCM Encoding Format ConfigurePPL AssignPPL Audit ConfigurePPL ConfigurePPL DeletePPL Timer ConfigureProduct License DownloadReset ConfigurationService State ConfigureSS7 CIC ConfigureSS7 ISUP Message Format ConfigureSS7 Signaling Link ConfigureSS7 Signaling Link Set ConfigureSS7 Signaling Route ConfigureSS7 Signaling Stack Configure
:	Checksum

Configuration Tag Behavior

The following table provides the tag configuration behavior on the CCS and Line cards when the host application sends an out-of-service action in the *Service State Configure* (0x000A) message. This table includes bearer and control channels/spans.

Card	CIC/B Channel T1/E1 Channel Out of Service	CIC/B Channel T1/E1 Span Out of Service	Link/DChannel T1/E1 Channel Out of Service	Link/DChannel T1/E1 Span Out of Service	VCICs Channel Out of Service	VCICs Span Out of Service
SS7 SS7/EXS Local Remote	SS7 - Tag Reset T1/E1 - NA	SS7 - NA T1/E1 - Tag Reset	SS7 - Tag Reset T1/E1 - NA	SS7 - NA T1/E1 - Tag Reset	SS7 - Tag Reset VT1/VE1 - NA	SS7 - NA VT1/VE1 - Tag Reset
ISDN	ISDN - Tag Reset T1/E1 - NA	ISDN - NA T1/E1 - Tag Reset	ISDN - Tag Reset T1/E1 - Tag Reset	ISDN - Tag Reset T1/E1 - Tag Reset	ISDN - Tag Reset VT1/E1 - NA	ISDN - NA VT1/VE1 - Tag Reset
CAS	SS7/ISDN - NA T1/E1 - Tag Reset	SS7/ISDN - NA T1/E1 - Tag Reset	Not Applicable	Not Applicable	Not Applicable	Not Applicable
DS3	SS7/ISDN - Tag Reset DS3 - Remains Same	SS7/ISDN - Tag Reset DS3 - Tag Reset	SS7/ISDN - Tag Reset DS3 - Remains same	SS7/ISDN - Tag Reset DS3 - Tag Reset	Not Applicable	Not Applicable

TCAPMessageRegister

Type SwitchKit API message

Description Use the *SK_TCAPMessageRegister* message to register an application for all TCAP-related PPL Event Indications given a specific Originating Point Code (OPC)

Important! Only events of Component Type 0x70 (TCAP TUSI) will be sent to the registered application.

Sent by Function `sk_pplTCAPRegister()`

Arguments The following table shows the arguments you can change:

Argument	Description
int OPC	The Originating Point Code of the SS7 TCAP stack generating the PPL Event Indications.

C Structure

```
typedef struct {
    int OPC;
    UBYTE RegisterFlag;
} SK_TCAPMessageRegister;
```

C Structure Response

```
typedef struct {
    int Status;
} SK_TCAPMessageRegisterAck;
```

C++ Class

```
class SKC_TCAPMessageRegister : public
    SKC_ToolkitMessage {
public:
    int getOPC() const;
    void setOPC(int x);
    UBYTE getRegisterFlag() const;
    void setRegisterFlag(UBYTE x);
};
```

C++ Class Response

```
class SKC_TCAPMessageRegisterAck : public SKC_ToolkitAck
{
public:
    int getStatus() const;
    void setStatus(int x);
};
```

TFTP Manage 0x003A

SwitchKit Name	TFTPManage
Type	EXS API and SwitchKit API message
Description	<p>TFTP Manage 0x003A</p> <p>This message is used to download system software files and to query the TFTP configuration. You can send this message no matter what state the CSP Matrix Series 3 Card is in.</p> <p>To download CSP software, send the Begin Download ICB as well as appropriate Data ICB subtypes to send the following information to the CSP:</p> <ul style="list-style-type: none"> • Local Matrix Configuration Filename • Adjacent Matrix Configuration Filename • Local Matrix Server IP Address • Adjacent Matrix Server IP Address • Load Timestamp <p>Send the Begin Download ICB whenever you assign a new TFTP configuration file that contains information about a new load.</p> <p>To query the TFTP configuration, send one of the Action ICB query subtypes. The response to a query includes the corresponding Data ICB subtype.</p> <p>The Timestamp ICB is required only if the timestamp is not in the configuration file.</p> <p>Save Options, Load Version & Load Filename are CSP-initiated ICBs in response to the Query ICBs.</p> <p>Adjacent Matrix ICBs are needed only if you have an adjacent matrix, but they can also be sent directly to the adjacent matrix, instead of through the local matrix.</p>

Initiated by Host

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE ICBCount;
    UBYTE ICBDData[252];
} XL_TftpManage;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE MoreStatus;
    UBYTE ICBCount;
    UBYTE ICBDData[249];
} XL_TftpManageAck;
```

C++ Class

```
class XLC_TftpManage : public XLC_OutboundMessage {  
public:  
    UBYTE getICBCount() const;  
    void setICBCount(UBYTE x);  
    const UBYTE *getICBData() const;  
    UBYTE *getICBData();  
    void setICBData(UBYTE *x);  
};
```

C++ Class Response

```
class XLC_TftpManageAck : public XLC_AcknowledgeMessage {  
public:  
    unsigned short getStatus() const;  
    void setStatus(unsigned short x);  
    UBYTE getMoreStatus() const;  
    void setMoreStatus(UBYTE x);  
    UBYTE getICBCount() const;  
    void setICBCount(UBYTE x);  
    const UBYTE *getICBData() const;  
    UBYTE *getICBData();  
    void setICBData(UBYTE *x);  
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x003A)	3, 4	Message Type (0x003A)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Number of ICBs to follow		
9	<p>ICB</p> <p>0x01 Action ICBs</p> <p>0x00* Begin Download</p> <p>0x01* Query Local Matrix Configuration Filename</p> <p>0x02* Query Local Matrix Server IP Address</p> <p>0x03* Query Adjacent Matrix Configuration Filename</p> <p>0x04* Query Adjacent Matrix Server IP Address</p> <p>0x05* Query Timestamp</p> <p>0x06* Query Save Options</p> <p>0x07 Query Load Version</p> <p>0x08 Query Load Filename</p> <p>* These Action ICBs do not contain data. The Data Length is 0x00 which you have to indicate.</p> <p>0x02 Data ICBs</p> <p>0x37 Local CSP Matrix Series 3 Card Configuration Filename</p> <p>0x38 Local Matrix Server IP Address</p> <p>0x39 Adjacent CSP Matrix Series 3 Card Configuration Filename</p> <p>0x3A Adjacent Matrix Server IP Address</p> <p>0x3B Timestamp</p> <p>0x3C Save Options</p> <p>0x3D Load Version</p> <p>0x3E Load Filename</p>	8, 9	<p>Status MSB (0x00)</p> <p>Status LSB:</p> <p>0x00 No Timestamp Available A timestamp has not been supplied in either the TFTP Manage message or the TFTP Configuration file.</p> <p>0x01 Two Timestamps The TFTP Manage API message and the TFTP Configuration file both contain a timestamp.</p> <p>0x02 Old Timestamp The supplied timestamp is older than the adjacent CSP 2000 Matrix Card load's timestamp, therefore; the load will be overwritten.</p> <p>0x03 Timestamp is Equal The supplied timestamp is equal to the adjacent CSP Matrix Series 3 Card load's timestamp, therefore the cards might have different loads.</p> <p>0x04 Incorrect ICB Type The incorrect ICB Type is indicated in the More Status field. (Response continued below.)</p>
:	Checksum		

0x05	Incorrect ICB Subtype The incorrect ICB Subtype is indicated in the More Status field.
0x06	Incorrect ICB Length Fixed length ICB has wrong length or variable length (filename) ICB has a zero length. The ICB Subtype with the incorrect length is indicated in the More Status field.
0x07	Invalid Message Length Overall message length is less than minimum or aborts ICB in the middle. The supplied length is indicated in the More Status field.
0x08	TFTP of Configuration File Failed The failure reason is indicated in the More Status field.
0x09	TFTP Load Validation Failed The Failure Reason is indicated in the More Status field. See Response Status 0x08 (TFTP of Configuration File Failed) for Failure Reason values.
0x0A	Load is From Wrong Release New load information is for a different load than the CSP is running. The supplied load number is indicated in the More Status field.
0x0B	Filename is greater than 127 characters. The length supplied length is indicated in the More Status field.
0x0C	Unknown Load Number The host-supplied load number is greater than maximum known loads. The supplied load number is indicated in the More Status field.
0x0D	Response Would Be Too Long The message response would be greater than 256 bytes. There are too many queries in the message.
0x0E	Configuration File Changes Save Option Without Begin Download ICB A new TFTP Configuration filename is not compatible with the current load configuration. The failing load number is indicated in the More Status field.

10	<p>More Status</p> <p><u>Failure Reasons</u> (same as <i>Alarm Status</i> for TFTP alarm [0x1E] reported in the <i>Alarm</i> message)</p> <p>0x00 No Status</p> <p>0x01 TFTP device detected a protocol error such as a non DATA packet being received or did not get expected message acknowledgment. Happens when a file is not found.</p> <p>0x02 TFTP device timed out. Happens when IP Address is bad.</p> <p>0x03 TFTP device is out of sync.</p> <p>0x04 TFTP device has no more free socket IDs. Happens when another TFTP was in progress but was aborted.</p> <p>0x05 TFTP device cannot use a channel number given to it.</p> <p>0x06 TFTP device has not been initialized.</p> <p>0x07 No server IP Address has been given.</p> <p>0x08 No filename has been given.</p> <p>0x09 No timestamp has been given.</p> <p>0x0A A TFTP Configuration or BRecord File line is too long.</p> <p>0x0B Too much data for internal buffer.</p> <p>0x0C TFTP Configuration File line has no '='.</p> <p>0x0D TFTP Configuration File line has no second '='.</p> <p>0x0E TFTP Configuration File timestamp is missing a parameter.</p> <p>0x0F Load filename is greater than 127 characters.</p> <p>0x10 CSP Matrix Series 3 Card Label in TFTP Configuration file not set to SAVE_LOAD_TRUE</p> <p>0x11 Insufficient RAM. Not enough RAM on CSP Matrix Series 3 Card to save all loads configured to be saved.</p> <p>0x12 BRecord File is out of sync. There is no 'B' starting the record.</p> <p>0x13 BRecord receiver is out of sync. Expected a different record type.</p> <p>0x14 Invalid BRecord Checksum</p> <p>0x15 Invalid BRecord Count</p> <p>0x16 Incorrect Number of Bytes in BRecord Load.</p> <p>0x17 Invalid Load Checksum</p> <p>0x18 Incorrect Load for Card</p> <p>0x19 Incorrect Software Version. Load is from a different software version than that on the CSP Matrix Series 3 Card.</p> <p>0x1A Timeout waiting for response from download state machine</p> <p>0x1B Denied by download state machine. Occurs if a traditional host download is in progress</p> <p>0x1C Download state machine started a traditional host load</p> <p>0x1D Timeout waiting for buffer. Occurs if download state machine has a problem</p> <p>0x1E Stopped by host starting another TFTP</p> <p>0x1F Configuration file has a new save option for an existing load</p> <p>0x20 Operating system general error</p> <p>0x21 Operating system IO error</p>
----	---

11	Number of ICBs to follow
12	<p>ICB</p> <p>Action ICBs</p> <ul style="list-style-type: none"> 0x00* Begin Download 0x01* Query Local CSP Matrix Series 3 Card Configuration Filename 0x02* Query Local CSP Matrix Series 3 Card Server IP Address 0x03* Query Adjacent CSP Matrix Series 3 Card Configuration Filename 0x04* Query Adjacent CSP Matrix Series 3 Card Server IP Address 0x05* Query Timestamp 0x06* Query Save Options 0x07 Query Load Version 0x08 Query Load Filename <p>* These Action ICBs do not contain data.</p> <p>Data ICBs</p> <ul style="list-style-type: none"> 0x37 Local CSP Matrix Series 3 Card Configuration Filename 0x38 Local Matrix Server IP Address 0x39 Adjacent CSP Matrix Series 3 Card Configuration Filename 0x3A Adjacent Matrix Server IP Address 0x3B Timestamp 0x3C Save Options 0x3D Load Version 0x3E Load Filename
:	Checksum

Example Message (Socket Log Output for SwitchKit)

Query Local Matrix Filename

The trace below shows the *TFTP Manage* message sent from the host to the CSP to query the local matrix configuration filename. The bold number is the ICB Subtype, Query Local Matrix Configuration Filename (0x01).

Host to CSP:

```
[00 09] [00 3A] 00 00 FF 01 01 01 00
```

The trace below shows the response from the CSP. The bold numbers are the local matrix configuration filename, contained in a Query Local Matrix Configuration Filename action ICB (0x01).

CSP to Host:

```
[00 26] [00 3A] 00 00 FF [00 10] 01 01 01 [1B
63 3A 5C 74 66 74 70 62 6F 6F 74 5C 74 66
74 70 5F 6E 6F 5F 74 73 2E 63 66 67 00]
```

Time Set 0x00B5

SwitchKit Name TimeSet

Type EXS API and SwitchKit API message

Description **Time Set 0x00B5**

This message allows the host to set the clock used by the CSP to timestamp fault log entries.



WARNING

It is imperative that you set the system time properly.

Sent by Host

SwitchKit Code **Configuration**

```
TimeSet (
    Node = integer,
    Month = integer,
    Day = integer,
    Year = integer,
    Hour = integer,
    Minute = integer,
    Second = integer);
```

C Structure

```
typedef struct {
    UBYTE Month;
    UBYTE Day;
    UBYTE Year;
    UBYTE Hour;
    UBYTE Minute;
    UBYTE Second;
} XL_TimeSet;
```

C++ Class

```
class XLC_TimeSet : public XLC_OutboundMessage {
public:
    UBYTE getMonth() const;
    void setMonth(UBYTE x);
    UBYTE getDay() const;
    void setDay(UBYTE x);
    UBYTE getYear() const;
    void setYear(UBYTE x);
    UBYTE getHour() const;
    void setHour(UBYTE x);
    UBYTE getMinute() const;
    void setMinute(UBYTE x);
```

```

UBYTE getSecond() const;
void setSecond(UBYTE x);
};
    
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x000B)	1, 2	Length (0x0007)
3, 4	Message Type (0x00B5)	3, 4	Message Type (0x00B5))
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Month	8, 9	Status MSB, LSB
9	Day	10	Checksum
10	Year Represented by the last two digits. For example, 1995 = 0x5F.		
11	Hour		
12	Minute		
13	Second		
14	Checksum		

Tone Configure 0x0031

SwitchKit Name	TransmitToneConfig
Type	EXS API and SwitchKit API message
Description	<p>Tone Configure 0x0031</p> <p>This message is used to configure a transmit tone or call progress receive tone. If the host duplicates transmit tones in the system, it is only necessary to do so once per DSP card.</p> <p>For more information on customizing tones and patterns, refer to the <i>API Developer's Guide: Overview, Adjusting Attributes of Call Progress Tones</i>.</p>
Sent by	Host

SwitchKit Code Configuration

```
TransmitToneConfig (
    Node = integer,
    UpdateFlag = integer,
    Reserved = integer,
    ToneType = integer,
    Action = integer,
    Data = byte array);
```

C Structure

```
typedef struct {
    UBYTE UpdateFlag;
    UBYTE Reserved;
    UBYTE ToneType;
    UBYTE Action;
    UBYTE Data[249];
} XL_TransmitToneConfig;
```

C++ Class

```
class XLC_TransmitToneConfig : public XLC_OutboundMessage
{
public:
    UBYTE getUpdateFlag() const;
    void setUpdateFlag(UBYTE x);
    UBYTE getReserved() const;
    void setReserved(UBYTE x);
    UBYTE getToneType() const;
    void setToneType(UBYTE x);
    UBYTE getAction() const;
    void setAction(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

Overview of message The following table provides an overview of this message. The table following it provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0031)	3, 4	Message Type (0x0031)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status MSB, LSB
		10	Checksum
8	Update All Flag		
9	Reserved		
10	Tone Type		
11	Action		
12	Data[0]		
:	Checksum		

EXS API Hex Format - Detailed

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0031)	3, 4	Message Type (0x0031)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status MSB (0x00) Status LSB: 0x01 Maximum frequencies per tone exceeded. This response is returned if the Frequency Counter in the <i>Tone Configure</i> message exceeds two. 0x02 Maximum frequencies per tone group exceeded. This response is returned if the number of frequencies in the Tone Group exceeds 16 (including possible new frequencies in the <i>Tone Configure</i> message). 0x03 Invalid frequency. This response is returned if any frequency in the <i>Tone Configure</i> message exceeds the theoretical limit (4000 Hz).
		10	Checksum
8	Update All Flag A configuration is "engaged" when it is used when the tone is transmitted. 0x00 Store configuration but do not engage it. This is the normal case for large configuration of parameters. Configuration will not be engaged until another message has this bit set to 1. 0x01 Engage configuration immediately This is usually set for the last configuration message so configuration engaging takes place only once and for all of the new configuration previously sent.		
9	Reserved		
10	Tone Type 0x01 Call Progress Transmit Tones 0x02 DTMF Transmit Tones 0x03 MFR1 Transmit Tones 0x04 MFR2 Forward Transmit Tones 0x05 MFR2 Backward Transmit Tones 0x06 Call Progress Receive Tones		

Tone Configure 0x0031

11	<p>Action</p> <ul style="list-style-type: none"> 0x01 Configure Call Progress Tones 0x02 Reserved 0x03 Global Update of dBm levels (Tone Types 2–5)
12	<p>Data[0]</p> <p>The value of the <i>Data</i> fields is dependent on the value of the <i>Action</i> field.</p> <p>0x01 Configure Call Progress Tones</p> <p>The following data uniquely identifies an existing tone or a new tone. All values must be entered redefining that tone if it exists already. The maximum number of frequencies per tone is limited to the DSP load configured to play the tone (2 or 4).</p> <p>Caution! Changing this tone changes every tone pattern referencing this tone.</p> <ul style="list-style-type: none"> Data[0] Tone ID being changed Data[1] Reserved Data[2] Number of Frequencies making up tone (1 or 2 for receive tone) Data[3] Reserved Data[4] Frequency Value[0] Hz, MSB Data[5] Frequency Value[0] Hz, LSB Data[6] Frequency dBm[0], MSB - must be 0 for call progress receive tones (0x06). For call progress transmit tones (0x01), specify power level. Data[7] Frequency dBm[0], LSB : Data[n++] Frequency Value[n] Hz, MSB Data[n++] Frequency Value[n] Hz, LSB Data[n++] Frequency dBm[n], MSB Data[n++] Frequency dBm[n], LSB <p>0x02 Reserved</p> <p>0x03 Global Update of dBm Level</p> <p>This action changes the power level of the transmit tones of the specified type. The update affects all tone generators of the specified type in the system. For Tone Type 2 (DTMF) the dBm level for both low-band and high-band frequency components must be specified. The other Tone Types require only one dBm level to be specified.</p> <p>The default dBm levels are as follows:</p> <ul style="list-style-type: none"> DTMF (low-band)-4.0 dBm DTMF (high-band)-4.0 dBm MFR1 -7.0 dBm MFR2 (forward)-9.0 dBm MFR2 (backward)-9.0 dBm <p>Data for Tone Type 2</p> <ul style="list-style-type: none"> Data[0] dBm level for low-band frequency component, MSB Data[1] dBm level for low-band frequency component, LSB Data[2] dBm level for high-band frequency component, MSB Data[3] dBm level for high-band frequency component, LSB <p>Data for Tone Types 3–5</p> <ul style="list-style-type: none"> Data[0] dBm level, MSB Data[1] dBm level, LSB
:	Checksum

Example Message—Global Update of dBm Level

The following is an example using the *Tone Configure* message to change the power levels of DTMF tones to: -4 dBm for low-band frequency, and -7 dBm for high-band frequency.

Byte	Message	Value
0	Message Frame	0xFE
1	Length, MSB	0x00
2	Length, LSB	0x0D
3	Message Type, MSB	0x00
4	Message Type, LSB	0x31
5	Reserved	0x00
6	Sequence Number	SN
7	Logical Node ID	0xFF
8	Update All Flag	0x01
9	Reserved	0x00
10	Tone Type	0x02
11	Action	0x03
12	Data[0]: dBm level for low-band frequency component, MSB	0xFF
13	Data[1]: dBm level for low-band frequency component, LSB	0xFC
14	Data[2]: dBm level for high-band frequency component, MSB	0xFF
15	Data[3]: dBm level for high-band frequency component, LSB	0xF9
16	Checksum	CS

Example Message—Configure Tone

The following example shows the message contents that would be sent to configure a specific tone to be outpulsed.

Configure Tone: Tone ID Number 4 (2 Frequencies: 440Hz, -13dBm; 480Hz, -13dBm)

Byte	Message	Value
0	Message Frame	0xFE
1, 2	Length	0x0015
3, 4	Message Type	0x0031
5	Reserved	0x00
6	Sequence Number	SN
7	Logical Node ID	0xFF
8	Update All Flag	0x01
9	Reserved	0x00
10	Tone Type	0x01
11	Action	0x01
12	Data[0]: Tone ID being changed	0x04
13	Data[1]: Reserved	0x00
14	Data[2]: Number of frequencies making up tone	0x02
15	Data[3]: Reserved	0xF9
16	Data[4]: Frequency Value[0] Hz, MSB	0x01
17	Data[5]: Frequency Value[0] Hz, LSB	0xB8

18	Data[6]: Frequency dBm[0], MSB	0xFF
19	Data[7]: Frequency dBm[0], LSB	0xF3
20	Data[8]: Frequency Value[1], Hz MSB	0x01
21	Data[9]: Frequency Value[1] Hz, LSB	0xE0
22	Data[10]: Frequency dBm[1], MSB	0xFF
23	Data[11]: Frequency dBm[1], LSB	0xF3
24	Checksum	CS

Configuring a Sample Pattern ID

The following example shows how to configure Pattern ID 0x17 to receive a three-second, 400 Hz tone. To perform this task, the host sends three API messages. First, the host sends the *Tone Configure* message to replace the default tone (Tone ID 0x0F, 425 Hz) with the new, 400 Hz tone. Second, the host sends the *CPA Pattern Configure* message to create the new Pattern ID 0x17. Third, the host sends the *CPA Class Configure* message to add Pattern 0x17 to the CPA class 0x00.

First Message: *Tone Configure*

Byte	Field Description
0	Message Frame 0xFE
1,2	Length (2 bytes) 0x0011
3,4	Message Type (2 bytes) 0x0031
5	Reserved 0x00
6	Sequence Number
7	Logical Node ID 0xFF
8	Update All Flag 0x01
9	Reserved 0x00
10	Tone Type 0x06
11	Action 0x01
12	Data[0] Tone ID being changed 0x0F
13	Data[1] Reserved 0x00
14	Data[2] Number of frequencies in tone 0x01
15	Data[3] Reserved 0x00
16	Data[4] Frequency Value[0], Hz, MSB 0x01
17	Data[5] Frequency Value[0], Hz, LSB 0x90
18	Data[6] Reserved 0x00
19	Data[7] Reserved 0x00
20	Checksum

Second Message: Call Progress Analysis Pattern Configure

Byte	Field Description
0	Message Frame 0xFE
1,2	Length (2 bytes) 0x0016
3,4	Message Type (2 bytes) 0x00B2
5	Reserved 0x00
6	Sequence Number
7	Logical Node ID 0xFF
8	Update All Flag 0x01
9	Pattern ID 0x17
10	Action: Add/Replace Pattern 0x01
11	Data[0] Pattern ID to report on detection 0x17
12	Data[1] Tone Group ID 0x00
13	Data[2] Pattern Configuration 0x01
14	Data[3] CPA Report on Pattern Loss 0x17
15	Data[4] Interval Cycles to Match 0x01
16	Data[5] Interval Cycles to Report 0x01
17	Data[6] Reserved 0x00
18	Data[7] Interval Descriptor Count 0x01
19	Data[8] Tone ID 0x0F
20	Data[9] Reserved 0x00
21	Data[10] Minimum filter (in 10 ms units) MSB 0x00
22	Data[11] Minimum filter (in 10 ms units) LSB 0x50
23	Data[12] Maximum filter (in 10 ms units) MSB 0x00
24	Data[13] Maximum filter (in 10 ms units) LSB 0x00
25	Checksum

Third Message: *Call Progress Analysis Class Configure*

Byte	Field Description
0	Message Frame 0xFE
1,2	Length (2 bytes) 0x0009
3,4	Message Type (2 bytes) 0x00B3
5	Reserved 0x00
6	Sequence Number
7	Logical Node ID 0xFF
8	Update All Flag 0x01
9	Class ID 0x00
10	Action 0x01
11	Data[0] Pattern ID 0x17
12	Checksum

Tone Query 0x005A

SwitchKit Name TransmitToneQuery

Type EXS API and SwitchKit API message

Description **Tone Query 0x005A**

This message is used to query the transmit and receive tones assigned to a DSP, as well as the frequency and power levels of the tones.

NOTE: A *Tone Query* message in the middle of several *Tone Configure* messages shows the configuration as updated even when the *Update All Flag* is not set.

Sent by Host Application

SwitchKit Code **C Structure**

```
typedef struct {
    UBYTE QueryEntity;
    UBYTE QueryValue[252];
} XL_TransmitToneQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE Data[251];
} XL_TransmitToneQueryAck;
```

C++ Class

```
class XLC_TransmitToneQuery : public XLC_OutboundMessage
{
public:
    UBYTE getQueryEntity() const;
    void setQueryEntity(UBYTE x);
    const UBYTE *getQueryValue() const;
    UBYTE *getQueryValue();
    void setQueryValue(UBYTE *x);
};
```

C++ Class Response

```
class XLC_TransmitToneQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x005A)	3, 4	Message Type (0x005A)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Query Entity	8, 9	Status (MSB, LSB)
9	Data[0]	10	Data[0]
:	:	:	:
:	Data[n]	:	Data[n]
:	Checksum	:	Checksum

**EXS API Hex Format -
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x00NN)
3, 4	Message Type (0x005A)	3, 4	Message Type (0x005A)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Query Entity 0x01 Call Progress Transmit Tone Frequency and Power (per Tone ID) 0x02 Tone IDs (per DSP) 0x03 Global dBm level (per Tone Type) 0x04 Call Progress Receive Tone Frequency and Power (per Tone ID)	8, 9	Status (MSB, LSB) 0x04 Tone Not Found Returned if the queried tone is not found in the CPA database.

<p>9</p> <p>Data[0] The value of the message <i>Data</i> fields are dependent on the <i>Query Entity</i>, as shown below:</p> <p>0x01 Call Progress Transmit Tone Frequency and Power per Tone ID Data[0] Tone ID Value (1–C) Data[1] 0x00 Data[2] 0x00</p> <p>0x02 Call Progress Transmit Tone IDs per DSP For the MFDSP and DSP-ONE Cards: Data[0] DSP card slot number (0–15) Data[1] SIMM number (0–3) Data[2] DSP number (0–3)</p> <p>0x03 Global dBm level per Tone Type Data[0] Tone Type (2–5) Data[1] 0x00 Data[2] 0x00</p>	<p>10</p> <p>Data[0] The value of the response <i>Data</i> fields are dependent on the <i>Entity</i>, as shown below.</p> <p>0x01 Frequency and Power Data[0] Entity Data[1] Tone ID Data[2] Number of Frequencies making up tone Data[3] Reserved Data[4] Frequency Value[0] Hz, MSB Data[5] Frequency Value[0] Hz, LSB Data[6] Frequency dBm[0], MSB Data[7] Frequency dBm[0], LSB : Data[...] Frequency Value[n] Hz, MSB Data[...] Frequency Value[n] Hz, LSB Data[...] Frequency dBm[n], MSB Data[n] Frequency dBm[n], LSB</p> <p>0x02 Call Progress Transmit Tone IDs Data[0] Entity Data[1] Number of Tone IDs supported per DSP Data[2] DSP Channel Number Data[3] Tone ID : Data[...] DSP Channel Number Data[n] Tone ID</p> <p>0x03 Global dBm Level The value of the data bytes depends on the tone type as shown below.</p> <p>For Tone Type 2 (DTMF) Data[0] Entity Data[1] Tone Type Data[2] dBm level for low-band frequency component, MSB Data[3] dBm level for low-band frequency component, LSB Data[4] dBm level for high-band frequency component, MSB Data[5] dBm level for high-band frequency component, LSB</p> <p>For Tone Types 3–5 Data[0] Entity Data[1] Tone Type Data[2] dBm level, MSB Data[3] dBm level, LSB</p>
--	--

Tone Query 0x005A

	<p>0x04 Call Progress Receive Tone Frequency and Threshold per Tone ID</p> <p>Data[0] Tone ID Value (0x0 - 0x11) Data[1] Tone Group (0x0 - 0x09) Data[2] Reserved</p>		<p>0x04 Call Progress Tone Receive Frequency and Threshold</p> <p>Data [0] Query entity Data [1] Tone ID (0x0 - 0x11) Data [2] Number of frequencies making up the tone (1 or 2) Data [3] Tone group Data [4] Frequency value [0] Hz, MSB Data [5] Frequency value [0] Hz, LSB Data [6] CPA receiver detection threshold, dBm, MSB Data [7] CPA receiver detection threshold, dBm, LSB : : Data [n++] Frequency value [n] Hz, MSB Data [n++] Frequency value [n] Hz, LSB Data [n++] CPA receiver detection threshold, dBm, MSB Data [n++] CPA receiver detection threshold, dBm, LSB</p>
:	Data[n]	:	Data[n]
:	Checksum	:	Checksum

Example: Query Call Progress Transmit Tone Frequency and Power

This is an example of a message to query Call Progress Transmit Tone Frequency and Power per Tone ID for Tone ID Number 4 configured in the example for the *Tone Configure* message.

Byte	Message Field	Value
0	Frame	0xFE
1, 2	Length	0x0009
3, 4	Message Type	0x005A
5	Reserved	0x00
6	Sequence Number	SN
7	Logical Node ID	0xFF
8	Query Entity	0x01
9	Data[0]: Tone ID Value	0x04
10	Data[1]	0x00
11	Data[2]	0x00
12	Checksum	CS

Byte	Response Field	Value
0	Message Frame	0xFE
1, 2	Length	0x0013
3, 4	Message Type	0x005A
5	Reserved	0x00
6	Sequence Number	SN
7	Logical Node ID	0xFF
8, 9	Status (MSB, LSB)	0x0010
10	Data[0]: Entity	0x01
11	Data[1]: Tone ID being queried	0x04
12	Data[2]: Number of frequencies making up tone	0x02
13	Data[3]: Reserved	0x00
14, 15	Data[4, 5]: Frequency Value[0] Hz (MSB, LSB)	0x01B8
16, 17	Data[6, 7]: Frequency dBm[0] (MSB, LSB)	0xFFFF3
18, 19	Data[8, 9]: Frequency Value[n] Hz (MSB, LSB)	0x010E
20, 21	Data[10, 11]: Frequency dBm[n] (MSB, LSB)	0xFFFF3
22	Checksum	CS

Example: Query Call Progress Transmit Tone IDs

This is an example of a message to query Call Progress Transmit Tone IDs.

Byte	Message Field	Value
0	Frame	0xFE
1, 2	Length	0x0009
3, 4	Message Type	0x005A

Tone Query 0x005A

5	Reserved	0x00
6	Sequence Number	SN
7	Logical Node ID	0xFF
8	Query Entity	0x02
9	Data[0]: MFDSP Card Slot Number	0x01
10	Data[1]: SIMM Number	0x02
11	Data[2]: DSP Number	0x03
12	Checksum	CS

Byte	Response Field	Value
0	Message Frame	0xFE
1, 2	Length	0x000N
3, 4	Message Type	0x005A
5	Reserved	0x00
6	Sequence Number	SN
7	Logical Node ID	0xFF
8, 9	Status (MSB, LSB)	0x0010
10	Data[0]: Entity	0x02
11	Data[1]: Number of Tone IDs Supported	0x0C
12	Data[2]: DSP Channel Number	0x0B
13	Data[3]: Tone ID Value	0x0C
:	:	:
N	Checksum	CS

TransferChanMsg

Type SwitchKit API message

Description Use the *SK_TransferChanMsg* message when an application requires a different application to receive messages associated with a channel.

Span and channel describe the channel that is to be transferred to the target application. This channel must already be assigned to the current application. The Tag and Data fields are passed along without modification. The target application receives *SK_TransferChanMsg* with all the fields entered as they were entered by the sending application. The target application is then responsible for returning the channel to the LLC when it is finished with it.

Important! The span and channel must be specified in both the *TransferChanMsg* and the *TransferChanMsgAck* in order for the LLC to properly route this message to the appropriate applications. The channel is not physically transferred to the new application until the requesting application receives the Ack.

The application that receives the message must send a *SK_TransferChanMsgAck* to let the sending application know that the message was received. This can be done with the following code:

```
int handleTransfer(SK_Event *evt, void *tag){
    MsgStruct *msg = evt->IncomingMsg;
    SK_MSG_SWITCH(msg) {
        CASE_TransferChanMsg(tcm) {
            SK_TransferChanMsgAck a;
            sk_initMsg(&a, TAG_TransferChanMsgAck);
            a.Status = 0x10;
            sk_setSeqNum(&a, sk_getSeqNum(tcm));
            sk_sendMsgStruct((MsgStruct *)&a, NULL, NULL);
        };
    };
};
```

Sent by Application

C Structure

```
typedef struct {
    unsigned short Span;
    UBYTE Channel;
    int Target;
    unsigned short AckTimeout;
    int Tag1;
    int Tag2;
```

```

int Tag3;
int Tag4;
char AppGroupTarget[16];
unsigned short DataSize;
UBYTE Data[210];
} SK_TransferChanMsg;

```

C Structure Response

```

typedef struct {
int Status;
unsigned short Span;
UBYTE Channel;
unsigned short AckTimeout;
int Tag1;
int Tag2;
int Tag3;
int Tag4;
unsigned short DataSize;
UBYTE Data[226];
} SK_TransferChanMsgAck;

```

C++ Class

```

class SKC_TransferChanMsg : public SKC_ToolkitMessage {
public:
unsigned short getSpan() const;
void setSpan(unsigned short x);
UBYTE getChannel() const;
void setChannel(UBYTE x);
int getTarget() const;
void setTarget(int x);
unsigned short getAckTimeout() const;
void setAckTimeout(unsigned short x);
int getTag1() const;
void setTag1(int x);
int getTag2() const;
void setTag2(int x);
int getTag3() const;
void setTag3(int x);
int getTag4() const;
void setTag4(int x);
const char *getAppGroupTarget() const;
void setAppGroupTarget(const char *x);
unsigned short getDataSize() const;
void setDataSize(unsigned short x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};

```

C++ Class Response

```

class SKC_TransferChanMsgAck : public SKC_ToolkitAck {
public:
int getStatus() const;
void setStatus(int x);
unsigned short getSpan() const;
void setSpan(unsigned short x);
UBYTE getChannel() const;
void setChannel(UBYTE x);
unsigned short getAckTimeout() const;
void setAckTimeout(unsigned short x);

```

```
int getTag1() const;
void setTag1(int x);
int getTag2() const;
void setTag2(int x);
int getTag3() const;
void setTag3(int x);
int getTag4() const;
void setTag4(int x);
unsigned short getDataSize() const;
void setDataSize(unsigned short x);
const UBYTE *getData() const;
UBYTE *getData();
void setData(UBYTE *x);
};
```

- SK_BAD_APP_NAME** Target application does not exist.
- SK_BAD_PARAMETER** Invalid parameter. This includes when the channel specified by span and channel is not allocated to the source application.
- SK_BAD_MESSAGE** LLC is unable to send the TransferCan Msg to the target application.

Transmit Cadence Pattern Configure 0x0030

SwitchKit Name TransmitCadencePatternConfig

Type EXS API and SwitchKit API message

Description **Transmit Cadence Pattern Configure 0x0030**

This message is used to allow the host to configure any transmit cadence pattern. The host can create the cadence and the duration of the tone needed. Each cycle block shown below represents two durations, either of which could be silence (Tone ID=0) or some valid Tone ID. These cycle blocks can be repeated or strung together with different cycle blocks. For tones which are to be continuously on the duration should be set to 0xFFFF.

Sent by Host

SwitchKit Code **Configuration**

```
TransmitCadencePatternConfig (
    Node = integer,
    UpdateFlag = integer,
    Action = integer,
    Data = byte array);
```

C Structure

```
typedef struct {
    UBYTE UpdateFlag;
    UBYTE Action;
    UBYTE Data[251];
} XL_TransmitCadencePatternConfig;
```

C++ Class

```
class XLC_TransmitCadencePatternConfig : public
    XLC_OutboundMessage {
public:
    UBYTE getUpdateFlag() const;
    void setUpdateFlag(UBYTE x);
    UBYTE getAction() const;
    void setAction(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0030)	3, 4	Message Type (0x0030)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status (MSB, LSB)
		10	Checksum
8	Update All Flag		
9	Action		
10	Data[0]		
:	:		
:	Checksum		

**EXS API Hex Format -
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0030)	3, 4	Message Type (0x0030)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
		8, 9	Status (MSB, LSB)
		10	Checksum
8	<p>Update All Flag A configuration is “engaged” when it is used when the tone is transmitted. 0x00Store configuration but do not engage it. This is the normal case for large configuration of parameters. Configuration will not be engaged until another message has this bit set to 1. 0x01Engage configuration immediately This is usually set for the last configuration message so configuration engaging takes place only once and for all of the new configuration previously sent.</p> <p>NOTE: A <i>Transmit Tone Query</i> message shows the configuration as updated even when the Update All flag is not set.</p>		

Transmit Cadence Pattern Configure 0x0030

9	Action 0x01Configure Pattern
10	<p>Data[0] The following data representation uniquely identifies a cadence pattern to be generated using the <i>Connect Tone Pattern</i> message. This message can be used for an existing pattern or a new one. The maximum number of cycle blocks in one pattern is three. The cycle block representation is a generic form in which all transmit cadence patterns could be created.</p> <p>Durations are expressed in 10 ms units.</p> <p>Data[0]Transmit Cadence Pattern ID Data[1]Number of Cycle Blocks Data[2]Block 1 Tone ID Data[3]Reserved Data[4]Block 1 On Duration, MSB Data[5]Block 1 On Duration, LSB Data[6]Block 1 Off Duration, MSB Data[7]Block 1 Off Duration, LSB Data[8]Number of times to execute this on and off cadence within this block, MSB Data[9]Number of times to execute this on and off cadence within this block, LSB : Data[n++]Block n Tone ID Data[n++]Reserved Data[n++]Block n On Duration, MSB Data[n++]Block n On Duration, LSB Data[n++]Block n Off Duration, MSB Data[n++]Block n Off Duration, LSB Data[n++]Number of times to execute this on and off cadence within this block, MSB Data[n++]Number of times to execute this on and off cadence within this block, LSB</p>
:	Checksum

Example

NOTE: To transmit the pattern continuously, do not set the “Number of times. . .” parameter in this message to 0xFFFF. Instead, set the appropriate parameter in the *Connect Tone Pattern* message. Please refer to the *Connect Tone Pattern* message for more information.

The following example shows the message contents that would be sent to configure a specific cadence pattern to be transmitted on any channel.
Durations are expressed in 10 ms units.

Configure Pattern: Pattern ID Number 2 (Ringback)

Cycle Block 1: Tone ID 4, 400 ms On/200 ms Off, execute once

Cycle Block 2: Tone ID 4, 400 ms On/3000 ms Off, execute once

Byte	Message	Value
0	Frame	0xFE
1	Length, MSB	0x00
2	Length, LSB	0x19
3	Message Type, MSB	0x00

Transmit Cadence Pattern Configure 0x0030

Byte	Message	Value
4	Message Type, LSB	0x30
5	Reserved	0x00
6	Sequence Number	SN
7	Logical Node ID	0xFF
8	Update All Flag	0x01
9	Action	0x01
10	Data[0]: Transmit Pattern ID (0x01-0x3F)	0x02
11	Data[1]: Number of Cycle Blocks	0x02
12	Data[2]: Block 1 Tone ID	0x04
13	Data[3]: Reserved	0x00
14	Data[4]: Block 1 On Duration, MSB	0x00
15	Data[5]: Block 1 On Duration, LSB	0x28
16	Data[6]: Block 1 Off Duration, MSB	0x00
17	Data[7]: Block 1 Off Duration, LSB	0x14
18	Data[8]: Number of times to execute this cycle, MSB	0x00
19	Data[9]: Number of times to execute this cycle, LSB	0x01
20	Data[10]: Block 2 Tone ID	0x04
21	Data[11]: Reserved	0x00
22	Data[12]: Block 2 On Duration, MSB	0x00
23	Data[13]: Block 2 On Duration, LSB	0x28
24	Data[14]: Block 2 Off Duration, MSB	0x01
25	Data[15]: Block 2 Off Duration, LSB	0x2C
26	Data[16]: Number of times to execute this cycle, MSB	0x00
27	Data[17]: Number of times to execute this cycle, LSB	0x01
28	Checksum	CS

Transmit Cadence Pattern Query 0x0059

SwitchKit Name	TransmitCadencePatternQuery
Type	EXS API and SwitchKit API message
Description	Transmit Cadence Pattern Query 0x0059 This message is used to query any of the configured transmit cadence parameters.
Sent by	Host
SwitchKit Code	C Structure

```
typedef struct {
    UBYTE QueryEntity;
    UBYTE Data;
} XL_TransmitCadencePatternQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE Data[251];
} XL_TransmitCadencePatternQueryAck;
```

C++ Class

```
class XL_TransmitCadencePatternQuery : public
    XLC_OutboundMessage {
public:
    UBYTE getQueryEntity() const;
    void setQueryEntity(UBYTE x);
    UBYTE getData() const;
    void setData(UBYTE x);
};
```

C++ Class Response

```
class XL_TransmitCadencePatternQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x0007)	1, 2	Length (0x00NN)
3, 4	Message Type (0x0059)	3, 4	Message Type (0x0059)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Query Entity 0x01 Transmit Cadence Pattern	8, 9	Status (MSB, LSB)
9	Data 0x01–0x3F Transmit Cadence Pattern ID (1–63)		
10	Checksum		
Response continued below.			
10	Data The following response data represents the pattern configured for the ID specified. Durations are expressed in 10 ms units. Data[0] Transmit Cadence Pattern ID Data[1] Number of Cycle Blocks Data[2] Block 1 Tone ID Data[3] Reserved Data[4, 5] Block 1 On Duration (MSB, LSB) Data[6, 7] Block 1 Off Duration (MSB, LSB) Data[8, 9] Number of times to execute this cycle (MSB, LSB) : Data[n++] Block n Tone ID Data[n++] Reserved Data[n++] Block n On Duration (MSB, LSB) Data[n++] Block n Off Duration (MSB, LSB) Data[n++] Number of times to execute this cycle (MSB, LSB)		
:	Data[n]		
:	Checksum		

Example This is an example of a message sent to query a Transmit Cadence Pattern. The pattern queried (ID Number 2) is the one sent in the example for the *Transmit Cadence Pattern Configure* message.

Byte	Message Field	Value
0	Frame	0xFE
1	Length, MSB	0x00
2	Length, LSB	0x07
3	Message Type, MSB	0x00
4	Message Type, LSB	0x59
5	Reserved	0x00
6	Sequence Number	SN
7	Logical Node ID	0xFF
8	Query Entity	0x01
9	Data[0]	0x02
10	Checksum	CS

Byte	Response Field	Value
0	Message Frame	0xFE
1	Length, MSB	0x00
2	Length, LSB	0x07
3	Message Type, MSB	0x00
4	Message Type, LSB	0x59
5	Reserved	0x00
6	Sequence Number	SN
7	Logical Node ID	0xFF
8	Status, MSB	0x00
9	Status, LSB	0x10
5	Data[0]: Transmit Pattern ID (1-63)	0x02
6	Data[1]: Number of Cycle Blocks	0x02
7	Data[2]: Block 1 Tone ID	0x04
8	Data[3]: Reserved	0x00
9	Data[4]: Block 1 On Duration, MSB	0x00
10	Data[5]: Block 1 On Duration, LSB	0x28
11	Data[6]: Block 1 Off Duration, MSB	0x00
12	Data[7]: Block 1 Off Duration, LSB	0x14
13	Data[8]: Number of times to execute this cycle, MSB	0x00
14	Data[9]: Number of times to execute this cycle, LSB	0x01
15	Data[10]: Block 2 Tone ID	0x04
16	Data[11]: Reserved	0x00
17	Data[12]: Block 2 On Duration, MSB	0x00
18	Data[13]: Block 2 On Duration, LSB	0x28
19	Data[14]: Block 2 Off Duration, MSB	0x01
20	Data[15]: Block 2 Off Duration, LSB	0x2C

Transmit Cadence Pattern Query 0x0059

21	Data[16]: Number of times to execute this cycle, MSB	0x00
22	Data[17]: Number of times to execute this cycle, LSB	0x01
23	Checksum	CS

Transmit Signaling Configure 0x0014

SwitchKit Name TransmitSignalingConfig

Type EXS API and SwitchKit API message

Description **Transmit Signaling Configure 0x0014**

This message is used to define the signaling transmitted by the CSP for non-standard trunk or line interfaces.

NOTE: Do not use this message without first contacting Dialogic Technical Support.

Sent by Host

SwitchKit Code **Configuration**

```
TransmitSignalingConfig (  
    Node = integer,  
    Range = StartSpan:StartChan - EndSpan:EndChan,  
    SignallingType = integer,  
    SignallingValue = integer,  
    TransmissionMode = integer);
```

C Structure

```
typedef struct {  
    unsigned short StartSpan;  
    UBYTE StartChannel;  
    unsigned short EndSpan;  
    UBYTE EndChannel;  
    UBYTE SignallingType;  
    UBYTE SignallingValue;  
    UBYTE TransmissionMode;  
} XL_TransmitSignalingConfig;
```

C++ Class

```
class XLC_TransmitSignalingConfig : public  
    XLC_ChanRangeMessage {  
public:  
    unsigned short getStartSpan() const;  
    void setStartSpan(unsigned short x);  
    UBYTE getStartChannel() const;  
    void setStartChannel(UBYTE x);  
    unsigned short getEndSpan() const;  
    void setEndSpan(unsigned short x);  
    UBYTE getEndChannel() const;  
    void setEndChannel(UBYTE x);  
    UBYTE getSignallingType() const;  
    void setSignallingType(UBYTE x);  
    UBYTE getSignallingValue() const;  
    void setSignallingValue(UBYTE x);  
    UBYTE getTransmissionMode() const;
```

```
void setTransmissionMode(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0014)	3, 4	Message Type (0x0014)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8, 9	Status (MSB, LSB)
:	Address Method 0x01 - Range	10	Checksum
	Number of AEs to follow		
	AEs 0x0D Channel(Starting) 0x0D Channel (Ending)		
:	Signaling Type 0x01 On-hook 0x02 Initial Outseize 0x03 Secondary Outseize 0x04 Inseize Acknowledgment 0x05 Answer 0x06 Post Outseize Acknowledgment		
:	Signaling Value This field is a bit mask. The meaning of each bit is listed below. The bit values are 0=Disabled, 1=Enabled. Bit 0 A Bit 1 B Bits 2-7 Reserved, must be set to 0		
:	Transmission Mode 0x01 A and B Fixed 0x02 A Fixed, B Pulsed 0x03 A Pulsed, B Fixed 0x04 A Pulsed, B Pulsed Pulsed mode timing is determined by ringing on/off timers.		
:	Checksum		

Example—Invert E&M Wink Start Signaling

This example shows the message sequence sent to invert the default E&M Wink Start transmit signaling on a channel. The default E&M signaling values are: on-hook: A = 0, B = 0; all other signaling: A = 1, B = 1.

A separate message must be sent to change each signaling type.

Signaling Type	0x01 (On-hook)
Signaling Value	0x03 (A = 1, B = 1)
Transmission Mode	0x01 (A and B fixed)

Signaling Type	0x02 (Initial Outseize)
Signaling Value	0x00 (A = 0, B = 0)
Transmission Mode	0x01 (A and B fixed)

Signaling Type	0x03 (Secondary Outseize)
Signaling Value	0x00 (A = 0, B = 0)
Transmission Mode	0x01 (A and B fixed)

Signaling Type	0x04 (Inseize Acknowledgment)
Signaling Value	0x00 (A = 0, B = 0)
Transmission Mode	0x01 (A and B fixed)

Signaling Type	0x05 (Answer)
Signaling Value	0x00 (A = 0, B = 0)
Transmission Mode	0x01 (A and B fixed)

Signaling Type	0x06 (Post Outseize Acknowledgment)
Signaling Value	0x00 (A = 0, B = 0)
Transmission Mode	0x01 (A and B fixed)

Trunk Type Configure 0x0011

SwitchKit Name TrunkTypeConfig

Type EXS API and SwitchKit API message

Description **Trunk Type Configure 0x0011**

The message is required only for the T-ONE card. It sets the trunk or line interface protocol for a channel or group of channels. Logical Span IDs must be assigned to channels before this message can be sent.

This should be the first message sent when configuring channels, as any configuration sent before it resets to the defaults for the trunk type specified.

Trunk types define the standard signaling protocol used by a channel or group of channels. The *Trunk Type Configure* message sets all of the channel configuration to the defaults associated with the trunk type (based on the AT&T publication 43801 and the Bellcore TR-TSY-000506 specifications).

The default trunk type is E&M for T1 spans, and PPL (ITU-T Compelled R2) for E1 spans. Trunk types may be different for each individual channel on a span. This allows applications that require mixed trunk types to reside on the same span.

Configuration associated with each trunk type include the start dial type, signal scanning filters and timers, and transmit signal timers. Any configuration which varies from the defaults must be reissued following the sending of a *Trunk Type Configure* message.

NOTE: This message cannot be used for channels on a T-ONE or E-ONE line card with the span format set to Clear Channel. If so, a NACK of 0x17 (Invalid Data Type) is returned.

Sent by Host

SwitchKit Code **Configuration**

```
TrunkTypeConfig (
    Node = integer,
    Range = StartSpan:StartChan - EndSpan:EndChan,
    TrunkType = integer);
```

C Structure

```
typedef struct {
    unsigned short StartSpan;
    UBYTE StartChannel;
    unsigned short EndSpan;
    UBYTE EndChannel;
    UBYTE TrunkType;
} XL_TrunkTypeConfig;
```

C++ Class

```
class XLC_TrunkTypeConfig : public XLC_ChanRangeMessage {
public:
    unsigned short getStartSpan() const;
    void setStartSpan(unsigned short x);
    UBYTE getStartChannel() const;
    void setStartChannel(UBYTE x);
    unsigned short getEndSpan() const;
    void setEndSpan(unsigned short x);
    UBYTE getEndChannel() const;
    void setEndChannel(UBYTE x);
    UBYTE getTrunkType() const;
    void setTrunkType(UBYTE x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x00NN)	1, 2	Length (0x0007)
3, 4	Message Type (0x0011)	3, 4	Message Type (0x0011)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	AIB	8, 9	Status (MSB, LSB)
:	Address Method 0x01 - Range	10	Checksum
	Number of AEs to follow		
	AEs 0x0D Channel (Starting) 0x0D Channel (Ending)		
:	Trunk Type 0x01 E&M 0x02 FXO Loopstart 0x03 FXS Loopstart 0x04 FXO Groundstart 0x05 FXS Groundstart 0x09 PPL (Programmable Protocol Language)		
:	Checksum		

Start Dial Signaling Defaults

The table below shows the start dial signaling defaults by trunk type:

TRUNK TYPE	START DIAL TYPE	
	Incoming	Outgoing
E&M	Wink	Wink
FXS-LS	Dialtone	None
FXS-GS	Dialtone	Fixed Pause
FXO-LS	None	Dialtone
FXO-GS	None	Dialtone
Dial Pulse Originating	Wink	Wink
Dial Pulse Terminating	Wink	Wink

UpgradeSoftwarePatch

TypeType	SwitchKit API message
Description	Use the <i>SK_UpgradeSoftwarePatch</i> message to upgrade node software specified by the functions setNode (UBYTE n) and getNode () of the base class <i>SKC_Message</i> .
Sent by	SwitchManager
Arguments	The following table shows the arguments that you can modify:

Argument	Description
Filename	File name of software patch to be downloaded.

C Structure	<pre>typedef struct { char Filename[230]; } SK_UpgradeSoftwarePatch;</pre>
C Structure Response	<pre>typedef struct { int Status; } SK_UpgradeSoftwarePatchAck;</pre>
C++ Class	<pre>class SKC_UpgradeSoftwarePatch : public SKC_ToolkitMessage { public: const char *getFilename() const; void setFilename(const char *x); };</pre>
C++ Class Response	<pre>class SKC_UpgradeSoftwarePatchAck : public SKC_ToolkitAck { public: int getStatus() const; void setStatus(int x); };</pre>

UserTimer

Type SwitchKit API message

Purpose Use the *SK_UserTimer* message to cause an event to arrive at a predetermined time. The message that arrives at timer expiration is a *SK_UserTimerAck*.

Description The timer is tracked in the lower layers of the API, not by the LLC. It uses the socket-select mechanism for determining when the timer has expired which allows the timers to be operating system-independent, efficient, and relatively accurate. Without a real-time operating system, there is no guarantee of timing.

There are no data fields in a *SK_UserTimer* message because, when sent with a tag through an appropriate toolkit sending function, a tag will also be associated with the acknowledgment that arrives after the time period has elapsed.

Sent by Application

How to Set Up a Timer in C In C, a variable of *SK_UserTimer* must be declared. The message must be initialized using *sk_initMsg()*. Then, the arguments must be initialized. Any of the SwitchKit send routines may be used to send the message and assign a message handler to handle the acknowledgement when the timer expires.

```
{
...
    SK_UserTimer userTimer;
    sk_initMsg(&userTimer, TAG_UserTimer);
    userTimer.Seconds = 5;
    // Number of seconds until timer expires.
    userTimer.GroupTag = 3;
    // Used to identify timer upon cancellation
    sk_sendMsgStruct((MsgStruct *)&userTimer, (void
    *)myData, timerExpiredHandler);
...
};
```

How to Set Up a Timer in C++

In C++, a object of class *SKC_UserTimer* is created. Then, the arguments must be initialized using the `setSeconds()` and `setGroupTag()` methods of the class. The `send` method, which is a member of a base class inherited by *SKC_UserTimer*, can be used to send the message.

```
{
...
    SKC_UserTimer userTimer;
    userTimer.setSeconds(5);
    // Number of seconds until timer expires
    userTimer.setGroupTag(3);
    // Used to identify timer upon cancellation
    userTimer.send((void *)myData, timerExpiredHandler);
...
};
```

Arguments

The following table shows the arguments you can change for *UserTimer*:

Argument	Description
Seconds	The delta time in seconds from when the <i>UserTimer</i> message is sent to when the <i>UserTimerAck</i> will be sent to indicate expiration of the timer.
GroupTag	Cancellation of a timer requires that GroupTag be set at creation of the timer. When you want to cancel a <i>UserTimer</i> , you need to send a <i>CancelUserTimer</i> message with the appropriate GroupTag. All currently unexpired UserTimer messages from that application with the corresponding GroupTag are canceled. It is important to realize that the GroupTag domain is unique to each application. This prevents one application from canceling another application's <i>UserTimer</i> . If multiple applications use the same GroupTag in a timer, and one application cancels that tag, only the timers from that application will be canceled. All other timers will still expire normally.

Return Status

The following table shows the possible return states:

Value	Description
SK_SUCCESS	Success
SK_CANCELED	Timer is cancelled (verify this)

C Structure

```
typedef struct {
    float Seconds;
    int GroupTag;
} SK_UserTimer;
```

C Structure Response

```
typedef struct {
    int Status;
} SK_UserTimerAck;
```

C++ Class

```
class SKC_UserTimer : public SKC_ToolkitMessage {
public:
    float getSeconds() const;
    void setSeconds(float x);
    int getGroupTag() const;
    void setGroupTag(int x);
};
```

C++ Class Response

```
class SKC_UserTimerAck : public SKC_ToolkitAck {
public:
    int getStatus() const;
    void setStatus(int x);
};
```

V5 Configuration Query 0x007C

SwitchKit Name	V5Query
Type	EXS API and SwitchKit API message
Description	<p>V5 Configuration Query 0x007C</p> <p>This message queries the configured values of all configurable V5.2 parameters.</p>
Sent by	Host
C Structure	<pre>typedef struct { UBYTE AddrInfo[30]; UBYTE QueryType; UBYTE Reserved; } XL_V5Query;</pre>
C Structure Response	<pre>typedef struct { unsigned short Status; UBYTE TLVCount; UBYTE Data[250]; } XL_V5QueryAck;</pre>
C++ Class	<pre>class XLC_V5Query : public XLC_OutboundMessage { public: const UBYTE *getAddrInfo() const; UBYTE *getAddrInfo(); void setAddrInfo(UBYTE *x); UBYTE getSlot() const; void setSlot(UBYTE x); XBYTE getV5ID() const; void setV5ID(XBYTE x); UBYTE getQueryType() const; void setQueryType(UBYTE x); UBYTE getReserved() const; void setReserved(UBYTE x); };</pre>
C++ Class Response	<pre>class XLC_V5QueryAck : public XLC_AcknowledgeMessage { public: unsigned short getStatus() const; void setStatus(unsigned short x); UBYTE getTLVCount() const; void setTLVCount(UBYTE x); const UBYTE *getData() const; UBYTE *getData(); void setData(UBYTE *x); };</pre>

EXS API Hex Format

MESSAGE		RESPONSE	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x007C)	3, 4	Message Type (0x007C)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Sequence Number (same as message)
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status (MSB, LSB) 0x01 Unrecognized Entity 0x02 Invalid C Channel 0x03 Unable to delete TLV from IMMSG 0x04 Unrecognized TLV tag 0x05 Incorrect AIB 0x06 Unable to add TLV to IMMSG 0x07 V5 IDs not configured for this slot 0x08 Reserved Type 0x09-0x0F Reserved Also refer to common Response Status Values
:	Address Method		
	Number of AEs to follow	10	Number of TLVs to follow
	AEs 0x01 Slot 0x2C V5 ID 0x2D V5 ID, Link ID		

V5 Configuration Query 0x007C

:	<p>Query Type</p> <p>0x01 Query all host-configured V5 IDs (uses Slot AIB). Lists TLVs that are configured per slot represented by the "V5 Create" TLV</p> <p>0x02 Query all host-configured C Channels per V5 ID. Lists TLVs represented by the "V5 C Channel Assignment" TLV</p> <p>0x03 Query all host-configured links per V5 ID. Lists TLVs represented by the "V5 Add Link" TLV</p> <p>0x04 Query all other host-configured options per V5 ID. TLVs from the V5 Configure message are used in the response. The TLVs returned for this entity are as follows:</p> <ul style="list-style-type: none"> • Variant ID (0x03) • Reserved type (0x08) • LE or AN side (0x05) • V5 Interface ID (0x10) <p>0x05 Query all host-configured user ports per V5 ID. Lists TLVs represented by the "Add V5.2 User Port Range" TLV. Shows all the configured user ports as ranges for a queried V5 ID. A maximum of 50 ranges can be returned in a single response.</p> <p>0x06 Query the Service State of the V5 ID. A tag of "FF" is reported in the response. The service state of the V5 ID is reported in the response to this query.</p> <p>0x07 Host-configured User Port Range. A TLV represented by the "Add V5.2 User Port Range" TLV will show one range of the lowest configured user port ID to the highest configured user port for the queried V5 ID.</p> <p>0x08 Query all configured Cpaths. Lists TLVs represented by the "V5 C Path Assignment" TLV.</p>	:	<p>TLVs</p> <p>0x0001 Create V5 ID 0x0003 V5 Variant ID 0x0004 V5 Variant Type 0x0005 V5 Side 0x0006 Add V5 User Port Range 0x0008 Reserved 0x0009 Reserved 0x000A V5 C Channel Assignment 0x000B V5 Cpath Assignment 0x000C V5 Add Link 0x000E Reserved 0x0010 V5 Interface ID</p>
:	Reserved (0x00)	:	
n+1	Checksum	n+1	Checksum

V5 Configure 0x007B

SwitchKit Name	V5Config
Type	EXS API and SwitchKit API message
Description	<p>V5 Configure 0x007B</p> <p>This message configures V5-specific options.</p>
Sent by	Host
SwitchKit Code	<p>Configuration</p> <pre>V5Config (ConfigType = integer, TLVCount = integer);</pre>

C Structure

```
typedef struct {
    UBYTE AddrInfo[30];
    UBYTE ConfigType;
    UBYTE TLVCount;
    UBYTE Data[221];
} XL_V5Config;
```

C++ Class

```
class XLC_V5Config : public XLC_OutboundMessage {
public:
    const UBYTE *getAddrInfo() const;
    UBYTE *getAddrInfo();
    void setAddrInfo(UBYTE *x);
    UBYTE getSlot() const;
    void setSlot(UBYTE x);
    XBYTE getV5ID() const;
    void setV5ID(XBYTE x);
    UBYTE getConfigType() const;
    void setConfigType(UBYTE x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE		RESPONSE	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0x0007)
3, 4	Message Type (0x007B)	3, 4	Message Type (0x007B)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Sequence Number
7	Logical Node ID	7	Logical Node ID
8 :	AIB Address Method Number of AEs to follow	8, 9	Status MSB, LSB (see details below table) 0x01 Invalid V5 Interface ID 0x02 Invalid TLV Length 0x03 Unrecognized TLV Tag 0x04 Invalid V5 Variant ID 0x05 Invalid V5 Variant Type 0x06 Invalid Configuration Value 0x07 Invalid User Port ID Range 0x08 Invalid Tag ID 0x09 V5 Interface is In Service 0x0A Invalid C Channel 0x0B Invalid Link ID 0x0D, 0x0E, 0x0F: Reserved
	AEs 0x01 Slot 0x2C V5 ID 0x2D V5 ID, Link ID	10	Checksum
:	Configure Type (0x01 = TLV data to follow)		
:	Number of V5 TLVs to follow		
	TLVs 0x0001 Create V5 ID 0x0002 Destroy V5 ID 0x0003 V5 Variant ID 0x0004 V5 Variant Type 0x0005 V5 Side 0x0006 Add V5 User Port Range 0x0007 Remove V5 User Port Range 0x0008 Reserved 0x0009 Reserved 0x000A V5 C Channel Assignment 0x000B V5 Cpath Assignment 0x000C V5 Add Link 0x000D V5 Remove Link 0x000E Reserved 0x000F V5 Remove C Channel Assignment 0x0010 V5 Interface ID 0x0012 Remove V5 Cpath Assignment		
n+1	Checksum		

Response Status Indications

The following response status indications are available for the V5 Configure message:

0x01 Invalid V5 Interface ID

Internal slot initialization error for the V5 ID, or the host may be trying to:

- assign a V5 interface ID when this ID is already assigned in the CSP for the given network side (AN or LE).
- create a V5 ID that is greater than the maximum allowed.
- destroy a V5 ID that is greater than the maximum allowed.
- add a C Channel to a V5 ID that is not configured in the system.
- add a C Channel to a V5 ID that is greater than the maximum allowed.
- create a V5 ID that already exists.

0x02 Invalid TLV Length

One or more TLVs used in this message have an incorrect length.

0x03 Unrecognized TLV Tag

Unrecognized internal IMSG TLV tag was retrieved from IMSG.

0x04 Invalid V5 Variant ID

A null pointer was encountered while attempting to configure the V5 variant ID for a V5 ID.

0x05 Invalid V5 Variant Type

A null pointer was encountered while attempting to configure the V5 variant type.

0x06 Invalid Configuration Value

- The host may be attempting to configure the network side as AN or LE and has passed an invalid value for the network side.
- A null pointer was configured while attempting to configure the network side.
- The host may have sent an invalid value for the Configure Type.

0x07 Invalid User Port ID Range

- The host may be attempting to add a user port range when there were not enough free user port entries in the configuration database to accommodate the range.
- The host may be attempting to add a user port range when a user port ID within the desired range was already configured.
- The host may be attempting to add or remove a user port range that was invalid. The low range must be less than or equal to the high range and the total range must be less than or equal to the maximum allowed for each card.

See next RSV (0x08) for more information.

0x08 Invalid Tag ID

- The host may be attempting to add or remove a user port ID that is greater than the maximum allowed.
- The host may be attempting to add or remove a user port range that contained an incorrect user port type field.
- The host may be attempting to add or remove more ranges than are allowed (50).
- The host may have encountered a null pointer when attempting to add or remove user ports.

0x09 V5 Interface is In Service

The host may be attempting to add or remove a C Channel for an interface that was in service.

0x0A Invalid C Channel

- The host may be attempting to add or remove a C Channel that was greater than the maximum allowed per interface.
- The host may be attempting to add a C Channel to a timeslot that was already being used.
- The host may be attempting to add a C Channel with an invalid C Channel type value.
- The host may be attempting to add a C Channel to an interface that already has this logical C Channel assigned to it.
- The host was attempting to remove a C channel with an invalid C Channel ID.

0x0B Invalid Link ID

The host may be trying to:

- add a link with a span ID that is already assigned to another V5 interface.
- add a link when the link ID is already assigned to this V5 ID.
- add or remove a link ID that is greater than the maximum allowed.
- add a C Channel with a link ID that is not configured.
- remove a link that has a C Channel on it.
- add a link containing a C Channel to an interface that is in service.
- add a C Channel with a span ID that does not belong to the corresponding link ID.
- add a C Channel with a link ID that is greater than the maximum allowed.
- remove a link that is not configured.

0x0C Duplicate User Port Assignment Attempt

- The host may be attempting to add a duplicate user port ID to a V5 ID for a different Timeslot for V5.1.
- A null pointer was encountered

Version Request 0x0002

SwitchKit Name	VersionRequestQuery
Type	EXS API and SwitchKit API message
Description	<p>Version Request 0x0002</p> <p>This message is used by the host to determine the versions of both the boot loader and system software. It also reports the timestamp for the system software of the CSP Matrix Series 3 Card(s).</p> <p>The host can also send this message to the standby CSP Matrix Series 3 Card.</p>
Sent by	Host

SwitchKit Code **C Structure**

```
typedef struct {
    BaseFields Base;
} XL_VersionRequestQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE BootMajor;
    UBYTE BootMinor;
    UBYTE SoftwareMajor;
    UBYTE SoftwareMinor;
    UBYTE SoftwareBuild;
    int MatrixStamp;
    int AdjMatrixStamp;
} XL_VersionRequestQueryAck;
```

C++ Class

```
class XLC_VersionRequestQuery : public
    XLC_OutboundMessage {
public:
    empty class
```

C++ Class Response

```
class XLC_VersionRequestQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getBootMajor() const;
    void setBootMajor(UBYTE x);
    UBYTE getBootMinor() const;
    void setBootMinor(UBYTE x);
    UBYTE getSoftwareMajor() const;
    void setSoftwareMajor(UBYTE x);
    UBYTE getSoftwareMinor() const;
```

```

void setSoftwareMinor(UBYTE x);
UBYTE getSoftwareBuild() const;
void setSoftwareBuild(UBYTE x);
int getMatrixStamp() const;
void setMatrixStamp(int x);
int getAdjMatrixStamp() const;
void setAdjMatrixStamp(int x);
};

```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0x0005)	1, 2	Length (0x0014)
3, 4	Message Type (0x0002)	3, 4	Message Type (0x0002)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	Checksum	8, 9	Status (MSB, LSB)
		10	Boot Loader Major Revision
		11	Boot Loader Minor Revision
		12	System Software Major Revision
		13	System Software Minor Revision
		14	System Software Build Number
		15-18	CSP Matrix Series 3 Card Timestamp (4 Bytes)
		19-22	Adjacent CSP Matrix Series 3 Card Timestamp (4 Bytes)
		23	Checksum

Notes:

1. If System Software Major Rev, System Software Minor Rev, and System Software build Number fields are “0,” system software is not present.
2. When in the process of downloading, the system software versions reflect the version of the code being downloaded. The version of the code which is being executed by the CSP may differ during download.
3. It is recommended that the host rely on the *Poll* message to determine when system software must be downloaded.
4. If a second CSP Matrix Series 3 Card is not installed, or if the system software is not present on the second CSP Matrix Series 3 Card, the *Adjacent Matrix Controller Timestamp* field will show “0.”
5. Timestamp equals number of seconds since January 1, 1970.

Virtual Card Configure 0x00E0

SwitchKit Name	VirtualSlotConfig
Type	EXS API and SwitchKit API message
Description	<p>Virtual Card Configure 0x00E0</p> <p>Use this message when configuring virtual spans to simulate the adding and removing of cards that do not exist in the system. The slot numbers you configure with this message are used when configuring the virtual spans. You must send this message prior to the <i>Assign Logical Span ID</i> message.</p> <p>NOTE: De-assigning all spans using the <i>Assign Logical Span ID</i> (0x00A8) message will clean up all virtual slots which acts as if the virtual cards were removed.</p>
Related Messages	Virtual Span Configure
Sent by	Host
Example Message	<p>The following example configures virtual slot 40 on board type 80, virtual VDAC card.</p> <pre>00 0d 00 E0 00 00 ff 00 00 01 01 01 02 40 80</pre>
SwitchKit Code	<p>Configuration</p> <pre>VirtualSlotConfig (Node = integer, ConfigType = integer, Data = byte array);</pre> <p>C Structure</p> <pre>typedef struct { BaseFields Base; UBYTE ConfigType; UBYTE Data[222]; } XL_VirtualSlotConfig;</pre> <p>C++ Class</p> <pre>class XLC_VirtualSlotConfig : public XLC_OutboundMessage { public: UBYTE getConfigType() const; void setConfigType(UBYTE x); const UBYTE *getData() const; UBYTE *getData(); void setData(UBYTE *x); };</pre>

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00E0)	3, 4	Message Type (0x00E0)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8, 9 :	AE 0x00 00 Null	8, 9	Status MSB (0x00) Status LSB: 0x01 Virtual Card Already Assigned. The slot has already been assigned a virtual card. 0x02 Slot Already Exists. The slot already contains a physical card. 0x03 No Available Timeslots Hardware has no more timeslots available. 0x10 Positive ACK Everything validated and action was initiated. 0x61 Invalid Slot Number. Slot Number is not 0x40-0x5F. 0x74 Invalid Card Type Slot has an incorrect type specified. 0x7F Software Module Locked System has not been configured with a Product License for Virtual Spans.
		10	Checksum
:	Configure Type 0x01 Use TLVs		
:	Number of TLVs to follow		
:	Data (TLVs) 0x01 Add Virtual Card 0x02 Remove Virtual Card		
:	Checksum		

Response Data The response data varies depending upon the value of the fields in the message.

Virtual Span Control 0x00E2

SwitchKit Name	VirtualSpanControl
Type	EXS API and SwitchKit API message
Description	<p>Virtual Span Control 0x00E2</p> <p>This message is used by the host to indicate to the CSP the Layer 1 state of a Virtual Span.</p>
Sent	Host
Example Message	<p>The following example brings all virtual spans in the node into service.</p> <p>00 0E 00 E2 00 00 FF 00 01 0C 02 00 00 00 01 00</p>

SwitchKit Code Configuration

```
VirtualSpanControl (
    Node = integer,
    Span = integer,
    Event = integer,
    Data = byte array);
```

C Structure

```
typedef struct {
    unsigned short Span;
    UBYTE reserved19[28];
    unsigned short Event;
    UBYTE Data[221];
} XL_VirtualSpanControl;
```

C++ Class

```
class XLC_VirtualSpanControl : public XLC_SpanMessage {
public:
    unsigned short getSpan() const;
    void setSpan(unsigned short x);
    unsigned short getEvent() const;
    void setEvent(unsigned short x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

EXS API Hex Format

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00E2)	3, 4	Message Type (0x00E2)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Same Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB 0x0012 Invalid Span Invalid Span was passed from host. 0x001D Invalid Event Type Host specified an invalid event type. 0x0063 Unassigned Span The logical span specified has not been assigned.
:	Address Method 0x01 Individual		
	Number of AEs to follow 0x01		
	AEs 0x0C Logical Span	10	Checksum
:	Event MSB, LSB 0x00 Span is Non-operational (same as Loss of Signal at Layer 1) 0x01 Span is Operational (same as No Loss of Signal at Layer 1)		
:	Data[0] ICB Count. This should always be sent from the host as 0x00.		
:	Checksum		

VoIP Protocol Configure 0x00EE

SwitchKit Name VOIPProtocolConfig

Type EXS API and SwitchKit API message

Description **VoIP Protocol Configure 0x00EE**

This message is used to configure H.323 and Session Initiation Protocol (SIP) software.

Sent by Host

Example Message (Socket Log Output for SwitchKit)

The following socket log output/example message shows local SIP registrations enabled:

```
00 0E 00 EE 00 00 FF 00 00 00 00 01 01 C8 01 04
```

Overview of message

The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00EE)	3, 4	Message Type (0x00EE)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB
:	Address Method 0x00 Individual AEs		
	Number of AEs to follow	10	Checksum
	AEs		
:	Number of TLVs to follow		
:	TLVs		
:	:		
:	Checksum		

SwitchKit Code Configuration

```
VOIPProtocolConfig (  
    Node = integer,  
    Slot = integer  
    TLVCount = integer,  
    Data = string);
```

NOTE: The Slot parameter is needed only when configuring the H.323 protocol. SIP is configured automatically by SwitchKit.

C Structure

```
typedef struct {  
    UBYTE AddrInfo[30];  
    UBYTE TLVCount;  
    UBYTE Data[222];  
} XL_VOIPProtocolConfig;
```

C++ Class

```
class XLC_VOIPProtocolConfig : public XLC_OutboundMessage  
{  
public:  
    const UBYTE *getAddrInfo() const ;  
    UBYTE *getAddrInfo();  
    void setAddrInfo(UBYTE *x);  
    UBYTE getSlot() const };  
    void setSlot(UBYTE x);  
    UBYTE getTLVCount() const;  
    void setTLVCount(UBYTE x);  
    const UBYTE *getData() const;  
    UBYTE *getData();  
    void setData(UBYTE *x);  
};
```

**EXS API Hex Format -
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00EE)	3, 4	Message Type (0x00EE)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8	Status MSB
:	Address Method	9	Status, LSB 0x01 INVALID_TLV_DATA 0x02 INVALID_DATA_TYPE 0x03 INVALID_NUMBER_OF_TLVs 0x04 INVALID_TLV_LENGTH 0x05 INVALID_PROTOCOL 0x06 INVALID_TLV 0x07 INVALID_PORT_NUMBER 0x08 INVALID_DNS 0x09 PROTOCOL RUNNING 0x0A PROTOCOL MISMATCH 0x0B INVALID_AIB 0x0C INVALID_STATE - Cannot configure the gateway alias names because the IP Signaling Series 3 card is already registered with a gatekeeper. 0x0D MANDATORY_TLVs_MISSING
	0x01 Individual AEs		
	Number of AEs to follow		
	0x01	10	Checksum
	AEs		
	0x00 00 00 00 Null (for SIP)		
	0x01 Slot (for H.323)		
	Number of TLVs to follow		

:	<p>TLVs</p> <p>Mandatory:</p> <ul style="list-style-type: none"> 0x01C8 Protocol Type (for H.323 and SIP) <p>Optional</p> <ul style="list-style-type: none"> 0x0006 Route Group ID 0x0008 Criteria Type 0x000F Router Protocol ID 0x0013 Routing Method 0x0115 Call Agent Mode 0x011C SIP Stack T.38 Fax Support 0x01AE Poll Interval Configure 0x01C8 Protocol Type 0x01C9 Primary Matrix IP Port 0x01CA Secondary Matrix IP Port 0x01D6 DNS IP Address 0x01D7 Default Domain Name 0x01ED RFC 2833 Jitter Buffer Size 0x01EE Invalid Packet Alarm Threshold 0x01EF Initial UDP Port Number 0x01F0 Number of UDP Ports 0x01F1 RFC 2833 Dynamic Payload Type 0x0262 Use SIP Local Port 0x0263 Use SIP Remote Port/Default Proxy Port 0x0264 Use SIP Remote Host/Default Proxy Host 0x0265 Use SIP Site ID 0x0266 Use SIP Anonymous Caller 0x0267 Use SIP Invite T1 Timer 0x0268 Use SIP BYE T1 Timer 0x0269 Use SIP T2 Timer 0x026A Use SIP Maximum Retransmissions Invite 0x026B Use SIP Max Retransmissions BYE 0x0270 SIP Tunnel Type 0x0272 Use SIP Registration Timeout 0x0277 SIP Version 0x0274 Use No Local Media 0x0275 Local Registration Enable/Disable 0x0279 SIP Local Route Lookup 0x027A SIP Registration Mode 0x027B Session Interval 0x027C Min-SE Interval 0x027D SIP Transport Type 0x027F SIP Message Information Mask 0x027E SIP Persistent Sockets 0x0280 SIP Existing Socket Reuse 0x0281 SIP Idle Socket Timeout 0x0282 PPL Event Notification Mask 0x02BC Gatekeeper IP and Port
---	--

VoIP Protocol Configure 0x00EE

	0x02BE Variant ID 0x02BF Version Number 0x02C0 Terminal Type 0x02C1 Vendor ID 0x02C2 Gatekeeper Auto Discovery* 0x02C3 Gatekeeper Auto Registration* 0x02C4 Gateway E164 0x02C6 Gateway URL ID 0x02C7 Gateway E-mail ID 0x02C8 H.225 T301 Timer 0x02C9 H.225 T303 Timer 0x02CA H.225 T310 Timer 0x02CB H.245 T101 Timer 0x02CC H.245 T103 Timer 0x02CD H.245 T105 Timer 0x02CE H.245 T106 Timer 0x02CF H.245 T108 Timer 0x02D0 H.245 T109 Timer 0x02D1 RAS GRQ Timer 0x02D2 RAS RRQ Timer 0x02D4 H.323 Message Information Mask 0x05EF Alarm Threshold Configure
:	Checksum

*For information on discovering and registering with a Gatekeeper, please refer to Discovering a Gatekeeper in the *Developer's Guide: Internet Protocol*.

VoIP Protocol Query 0x00EF

SwitchKit Name	VOIPProtocolQuery
Type	EXS API and SwitchKit API message
Description	VoIP Protocol Query 0x00EF Use this message to query the VoIP configuration.
Sent by	Host
SwitchKit Code	C Structure

```
typedef struct {
    UBYTE TLVCount;
    UBYTE Data[222];
} XL_VOIPProtocolQuery;
```

C Structure Response

```
typedef struct {
    unsigned short Status;
    UBYTE TLVCount;
    UBYTE Data[250];
} XL_VOIPProtocolQueryAck;
```

C++ Class

```
class XLC_VOIPProtocolQuery : public XLC_OutboundMessage
{
public:
    UBYTE getTLVCount() const
    void setTLVCount(UBYTE x)
    const UBYTE *getData() const
    UBYTE *getData()
    void setData(UBYTE *x);
};
```

C++ Class Response

```
class XLC_VOIPProtocolQueryAck : public
    XLC_AcknowledgeMessage {
public:
    unsigned short getStatus() const;
    void setStatus(unsigned short x);
    UBYTE getTLVCount() const;
    void setTLVCount(UBYTE x);
    const UBYTE *getData() const;
    UBYTE *getData();
    void setData(UBYTE *x);
};
```

Overview of message The following table provides an overview of this message. The table following it, provides the detail for each byte.

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00EF)	3, 4	Message Type (0x00EF)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Sequence Number
7	Logical Node ID	7	Logical Node ID
8	<u>AIB</u>	8, 9	Status MSB, LSB
:	Address Method 0x01		
	Number of AEs to follow	10	Checksum
	AEs		
:	Number of TLVs to follow		
:	TLVs		
:	:		
:	Checksum		

**EXS API Hex Format -
Detailed**

MESSAGE (White)		RESPONSE (Gray)	
Byte	Field Description	Byte	Field Description
0	Frame (0xFE)	0	Frame (0xFE)
1, 2	Length (0xNNNN)	1, 2	Length (0xNNNN)
3, 4	Message Type (0x00EF)	3, 4	Message Type (0x00EF)
5	Reserved (0x00)	5	Reserved (0x00)
6	Sequence Number	6	Sequence Number
7	Logical Node ID	7	Logical Node ID
8-11	<u>AIB</u> Address Method 0x00 - Individual AEs	8	Status MSB
	Number of AEs to follow	9	Status LSB
	AE 0x00 00 00 00 Null (for SIP) 0x01 Slot (for H.323)		0x01 INVALID_TLV_DATA 0x02 INVALID_DATA_TYPE 0x04 INVALID_TLV_LENGTH 0x05 INVALID_PROTOCOL 0x06 INVALID_TLV
12	Number of TLVs to follow	10	Number of TLVs to follow
		:	TLV (same as message)
		:	Checksum

13	<p>TLVs</p> <p>Mandatory:</p> <ul style="list-style-type: none"> 0x01C8 Protocol Type (for H.323 and SIP) <p>Optional</p> <ul style="list-style-type: none"> 0x0006 Route Group ID 0x0008 Criteria Type 0x000F Router Protocol ID 0x0013 Routing Method 0x0115 Call Agent Mode 0x011C SIP Stack T.38 Fax Support 0x01AE Poll Interval Configure 0x01C8 Protocol Type 0x01C9 Primary Matrix IP Port 0x01CA Secondary Matrix IP Port 0x01D6 DNS IP Address 0x01D7 Default Domain Name 0x01ED RFC 2833 Jitter Buffer Size 0x01EE Invalid Packet Alarm Threshold 0x01EF Initial UDP Port Number 0x01F0 Number of UDP Ports 0x01F1 RFC 2833 Dynamic Payload Type 0x0262 Use SIP Local Port 0x0263 Use SIP Remote Port/Default Proxy Port 0x0264 Use SIP Remote Host/Default Proxy Host 0x0265 Use SIP Site ID 0x0266 Use SIP Anonymous Caller 0x0267 Use SIP Invite T1 Timer 0x0268 Use SIP BYE T1 Timer 0x0269 Use SIP T2 Timer 0x026A Use SIP Maximum Retransmissions Invite 0x026B Use SIP Max Retransmissions BYE 0x0270 SIP Tunnel Type 0x0272 Use SIP Registration Timeout 0x0277 SIP Version 0x0274 Use No Local Media 0x0275 Local Registration Enable/Disable 0x0279 SIP Local Route Lookup 0x027A SIP Registration Mode 0x027B Session Interval 0x027C Min-SE Interval 0x027D SIP Transport Type 0x027F SIP Message Information Mask 0x027E SIP Persistent Sockets 0x0280 SIP Existing Socket Reuse 0x0281 SIP Idle Socket Timeout 0x0282 PPL Event Notification Mask 0x02BC Gatekeeper IP and Port 0x02D4 H.323 Message Information Mask
----	---

Value (Hex)	Name	Description
0x01	Invalid query type	User requested an incorrect query type
0x02	IMSG Error	Error due to IMSG operation, addTLV, getTLV, deleteTLV, addorReplace TLV.
0x03	Internal Message Error	Unable to send or create internal message
0x04	PSOS Error	Unable to get memory

*For information on discovering and registering with a Gatekeeper, please refer to "Discovering a Gatekeeper" in the *Developer's Guide: Internet Protocol*.

WatchChanGroup

Type	SwitchKit API message
Description	This message is used if your application sends the sk_watchChannelGroup() function.
Sent by	Function sk_watchChannelGroup() ;
C Structure	<pre>typedef struct { } SK_WatchChanGroup;</pre>
C++ Class	<pre>class SKC_WatchChanGroup : public SKC_AdminMessage { public: };</pre>

WriteCfgFile

Type	SwitchKit API message
Purpose	Use the <i>SK_WriteCfgFile</i> message to instruct SwitchManager to write the configuration file.
Description	When this message is received by SwitchManager, it will search through its database of messages for each slot on each node. Each message will be written to the new file.
Sent by	Application
Arguments	The following table shows the user modifiable arguments:

Argument	Description
Filename	The name of the file to write to. If blank (e.g. ""), then SwitchManager will write to a filename derived from the input file. For example, if the original file is myConfig.cfg and no filename is provided to <i>SK_WriteCfg</i> , the first time <i>SK_WriteCfg</i> is called, the output name will be myConfig_1.cfg.

C Structure	<pre>typedef struct { char Filename[230]; } <i>SK_WriteCfgFile</i>;</pre>
C Structure Response	<pre>typedef struct { int Status; } <i>SK_WriteCfgFileAck</i>;</pre>
C++ Class	<pre>class <i>SKC_WriteCfgFile</i> : public SKC_ToolkitMessage { public: const char *getFilename() const; void setFilename(const char *x); };</pre>
C++ Class Response	<pre>class <i>SKC_WriteCfgFileAck</i> : public SKC_ToolkitAck { public: int getStatus() const; void setStatus(int x); const UBYTE *getReserved() const; UBYTE *getReserved(); void setReserved(UBYTE *x); };</pre>

2 Address Elements

Overview

Purpose This chapter lists the Address Elements (AEs) used in the EXS API. The single-byte (0xNN) AEs are first, listed in hexadecimal numerical order. They are followed by the double-byte (0xNNNN) AEs, also listed in hexadecimal numerical order.

Introduction to Address Elements and Address Information Blocks

Generic AIB Format An Address Information Block (AIB) is used to address system objects, such as spans and channels. An AIB has a header, followed by one or more Address Elements. An Address Element is a group of data bytes that represent specific address information.

AIB Field Description
Address Method
Number of Address Elements
Address Element Type
Data Length (n)
Data[0]
:
Data[n-1]

Address Method

Use this field to specify one of two methods for the AIB:

Individual or Range

0x00 - Individual AEs

The Individual AEs method treats each Address Element as a single entity. Use the Individual AEs method to address the following:

- A single component, such as a channel or a slot
- Multiple independent components that are not in a range, such as two channels in a *Connect* message, where Address Element 1 defines Channel A and Address Element 2 defines Channel B.

0x01 - Range

Use this method to define the two Address Elements that are the starting and ending entities in a range.

For example, you use this method in an *Impulsing Parameters Configure* message to configure a range of channels, where Address Element 1 defines the starting channel and Address Element 2 defines the ending channel.

Important! The field is contained in the AIB in the API messages not in the AE tables in this chapter.

Number of Address Elements

This field indicates the number of Address Elements in the AIB.

An Address Element is a group of data bytes that describe an address. The number of data bytes in an Address Element varies, depending on the address element type.

Important! The field is contained in the AIB in the API messages not in the AE tables in this chapter.

Address Element (AE) Type

This field specifies the Address Element (AE) type. An AIB can contain several AEs. PPL Components use Address Elements in various PPL Messages.

If you use the *Individual AEs* Addressing Method, you can insert multiple AEs, of different types, into a single AIB.

If you use the *Range* Addressing Method, you can insert two, and only two, Address Elements, and they must both be of the same Address Type.

Data Length

This field indicates the number of data bytes in the AE immediately following this byte. If an AE contains no data, the value of this field is 0x00.

Data[0] . . . Data[n]

The number of Address Elements in an AIB depends on the message, address type, and addressing requirements.

For example, the table below shows an AIB that has one Address Element, which in turn contains two data fields.

Byte	AE Field Description
2	Address Element Type (0x12)
3	Data Length (0x02)
4	Data[0] – Slot Number
5	Data[1] – SIMM Number

Using AIBs

To use AIBs in a message, perform the following steps:

1. Determine the required address method and address type. Each message shows the AIBs that are supported by that message. The AIBs are presented in tables similar to the one above. Remember to build flexibility into your applications to allow for future expansion of the API, including new AIB Address Methods and Types, new addressing requirements, and new functionality.
2. Insert the information, as defined in the table, into the AIB section of the message. The addressing requirements of the message determine the values for the *Address Method* and *Number of Address Elements* fields. The object being addressed determines the address data fields.

The table below shows how an AIB is inserted into an API message. The address data fields are bytes 12–13.

MESSAGE	
Byte	Field Description
0	Frame Character (0xFE)
1	Length, MSB (0x00)
2	Length, LSB (0x0D)
3	Message Type, MSB
4	Message Type, LSB
5	Reserved (0x00)
6	Sequence Number
7	Node ID
8	AIB ..
9	
10	
11	
12	
13	
14	Checksum

Address Method
Number of Address Elements
Address Element Type
Data Length (0x04)
Data[0] – Slot Number
Data[1] – SIMM Number

Example 1

The table below shows the *ARP Cache Query* message, with an AIB inside it. You would refer to the Slot AE for its format.

MESSAGE (White)	
Byte	Field Description
0	Frame (0xFE)
1, 2	Length (0xNNNN)
3, 4	Message Type (0x00FC)
5	Reserved (0x00)
6	Sequence Number
7	Logical Node ID
:	AIB
	Address Method 0x00 - Individual AEs
	Number of AEs to follow
	AE 0x01 Slot
:	Data Type (0x00: TLVs)
:	Number of TLVs to follow
:	Data (TLVs) 0x01DC VoIP Module 0x01DD Flush ARP Cache Table Entry
:	Checksum

Example 2 - Individual AEs Method – Two Address Elements

The table below shows an AIB that uses the Individual AEs address method and that has two Address Elements. This AIB would be used in the *Connect* message, where:

- Address Element 1 defines Channel A
- Address Element 2 defines Channel B

MESSAGE (White)	
Byte	Field Description
0	Frame (0xFE)
1, 2	Length (0x0011)
3, 4	Message Type (0x0000)
5	Reserved (0x00)
6	Sequence Number
7	Logical Node ID

:	<u>AIB</u> Address Method 0x00 - Individual
	Number of AEs to follow 0x02
	AEs 0x0D Channel A 0x0D Channel B
:	Checksum

Example 2 - Range Method

The table below shows an example of an AIB that uses the Range address method. This AIB would be used in configuration messages to address a range of channels or other components, where:

- Address Element 1 defines the Starting Channel
- Address Element 2 defines the Ending Channel.

A range must contain **two and only two** Address Elements.

Important! To address a single channel using the range method, enter the same channel as both the starting and ending channel.

MESSAGE (White)	
Byte	Field Description
0	Frame (0xFE)
1, 2	Length (0xNNNN)
3, 4	Message Type (0x00D3)
5	Reserved (0x00)
6	Sequence Number
7	Logical Node ID
:	<u>AIB</u> Address Method 0x00 - Individual AEs
	Number of AEs to follow
	AEs 0x0D Channel (Starting) 0x0D Channel (Ending)
:	Flag 0x00 Busy Out Disabled 0x01 Busy Out Enabled

Address Elements

0x00 00 00 00 Null

Byte	AIB Field Description
2	Address Element: 0x00
3	Data Length 0x00

0x00 00 Null

Used in the *Matrix Configure*, *Matrix Query*, *System Configuration Query*, and *Virtual Card Configure* messages.

Byte	AIB Field Description
	There are no additional fields for this AE.

0x01 Slot

Byte	AIB Field Description
2	Address Element: 0x01
3	Data Length 0x01
4	Data[0] Slot Number <u>Front Of Chassis</u> 0x00–0x0F Line Card Slots 0–15 0x10 Front Fan Tray 0x11 Power2/PSC2 0x12 Power1/PSC1 0x16 Fan Tray 0x20 CSP Matrix Series 3 Card for CPU1 0x21 CSP Matrix Series 3 Card for CPU2 <u>Back Of Chassis</u> 0x13 Midplane 0x14 CSP Matrix Series 3 Card I/O for CPU1 0x15 CSP Matrix Series 3 Card I/O for CPU2 0x16 Rear Fan Tray 0x30–0x3F Line Card I/Os

* Some API messages such as the *Line Card Switchover* message require two AEs as follows:

Byte	AIB Field Description
2	Address Element: 0x01
3	Data Length 0x01
4	Data[0] Starting Slot Number

Byte	AIB Field Description
5	Address Element 0x01
6	Data Length 0x01
7	Data[0] Ending Slot Number

Slot Numbering

The tables below show the slot numbering schemes for the front and back of the CSP chassis.

Front Slot Numbering, CSP

Front Slots	Power Cards	Line Cards								Matrix Cards	Fan Tray
Chassis Label	Power2	15	14	13	12	11	10	9	8	CPU2	0x16
Software (Hex)	0x11	0x0F	0x0E	0x0D	0x0C	0x0B	0x0A	0x09	0x08	0x21	0x10
Chassis Label	Power1	7	6	5	4	3	2	1	0	CPU1	
Software (Hex)	0x12	0x07	0x06	0x05	0x04	0x03	0x02	0x01	0x00	0x20	

Back Slot Numbering, CSP

Back Slots	Matrix I/Os	Redundant or Standby Line Card I/Os							
Chassis Label	CPU1	0	1	2	3	4	5	6	7
Software (Hex)	0x14	0x30	0x31	0x32	0x33	0x34	0x35	0x36	0x37
Chassis Label	CPU2	8	9	10	11	12	13	14	15
Software (Hex)	0x15	0x38	0x39	0x3A	0x3B	0x3C	0x3D	0x3E	0x3F

0x06 ISDN Call Reference

Byte	AIB Field Description
2	Address Element: 0x06
3	Data Length: 0x04
4	Data[0] Slot Number
5	Data[1] Profile
6	Data[2] Subrate
7	Data[3] Call Reference

0x08 SS7 Stack

Byte	AIB Field Description
2	Address Element: 0x08
3	Data Length: 0x01
4	Data[0] Stack ID

0x09 SS7 Link

To address an SS7 signaling link, use the SS7 Link Address Element:

Byte	AIB Field Description
2	Address Element: 0x09
3	Data Length: 0x02
4	Data[0] Stack ID
5	Data[1] Link ID

0x0C Logical Span

Byte	AIB Field Description	
2	Address Element:	0x0C
3	Data Length:	0x02
4, 5	Data[0, 1]	Logical Span ID (MSB, LSB)

0x0D Channel

Byte	AIB Field Description	
2	Address Element:	0x0D
3	Data Length	0x03
4, 5	Data[0, 1]	Logical Span ID (MSB, LSB)
6	Data[2]	Channel

0x0E EXS Host-Slave Node

Byte	AIB Field Description	
2	Address Element	0x0E
3	Data Length	0x01
4	Data[0]	Logical Node ID (Host Node: 0x000 - 0x1F)
5	Address Element	0x0E
6	Data Length	0x01
7	Data[0]	Logical Node ID (Slave Node: 0x000 - 0x1F)

0x10 Logical/Physical Node

Byte	AIB Field Description
2	Address Element 0x10
3	Data Length: 0x05
4-7	<p>Data[0-3] Physical Node ID (4 Bytes)</p> <p>Enter the hexadecimal value of the last four digits of the serial number on the chassis or EXNET-ONE card. For the CSP 2090 and CSP 2040, you can find the last four digits on the rear of the CSP.</p> <p>EXNET Connect® Because EXNET Connect® is a card within a PC chassis, you assign the physical node ID by:</p> <p>Using the serial number of the EXNET Connect® card for the last four digits of the serial number.</p> <p>Set the two most significant bytes to 0x0001.</p> <p>If the serial number of the user's EXNET Connect® card was 0x03AD, in the Assign Logical Node ID message (0x10) define the value of the Physical Node ID field as 0x000103AD.</p>
8	<p>Data[4] Logical Node ID</p> <p>Enter the ID you are assigning to the node. The default logical node ID of any CSP is 0xFF.</p> <p>When you use message 0x0010 to assign a logical node ID to a node, you must use the value 0xFF in the Logical Node ID field in the message header. The new logical ID is the value of Byte 8 in the AIB.</p> <p>To reassign a logical node ID, you must bring the EXNET® ring out of service.</p>

0x11 Logical/Physical Span

Byte	AIB Field Description
2	Address Element 0x11
3	Data Length 0x04

Byte	AIB Field Description	
4,5	Data[0, 1]	Logical Span ID (MSB, LSB) Single-node system: 0-83 (0x0000-0x0053) Multi-node system: 0-1343 (0x0000-0x0053F) NOTE: When configuring a node for greater than 64 spans, please contact Dialogic Technical Support for assistance. The value 0xFFFF is reserved for de-assigning span IDs.
6	Data[2]	Slot For the CSP 2090: 0-15 For the CSP 2040: 0-3
7	Data[3]	Physical Span/Offset 4-Span Line Card: 0-3 8-Span Line Card: 0-7 16-Span Line Card: 0-15 DS3 Card: 0-27 VDAC-ONE Card 0-4 IPN Series 2 Profile 1 0-31 Profile 2 0-15

0x12 DSP SIMM

Byte	AIB Field Description	
2	Address Element 0x12	
3	Data Length	0x02
4	Data[0]	Slot Number (0x00-0x0F)
5	Data[1]	SIMM Number (0x00-0x03)

0x13 Physical Span

Byte	AIB Field Description	
2	Address Element 0x13	
3	Data Length	0x02
4	Data[0]	Slot Number (0x00-0x0F)
5	Data[1]	Physical Span
NOTE: When you use this AIB for Loop Timing on the EXNET-ONE or EXNET Connect® card, this field represents the span offset number of the span to be the loop timing source. To derive timing from the EXNET® ring, the Physical Span Number is not used.		

0x14 SS7 CIC Group

In this AIB, the fields define a CIC Group by assigning a span/channel as the Base CIC for the group, and by defining the number of channels in the group. The remaining channels in the group are assigned CIC numbers in sequence. All the CICs in a group must be on the same span.

Byte	AIB Field Description	
2	Address Element:	0x14
3	Data Length:	0x07
4	Data[0]	Stack ID
5, 6	Data[1, 2]	Base CIC Number (MSB, LSB) A number assigned by the host to the base CIC for the group of circuits (2 bytes).
7, 8	Data[3, 4]	Base CIC Span (MSB, LSB) Logical Span ID on which the channel to be assigned as the base CIC for the group is located. .
9	Data[5]	Base CIC Channel Channel to be assigned as the base CIC for the group (0x00–0x1F).
10	Data[6]	Number of CICs in Group The number of channels being assigned in the group (0x00–0x1F).

0x15 ISDN Primary D Channel

Byte	AIB Field Description
2	Address Element: 0x15
3	Data Length: 0x04
4	Data[0] Slot Number
5, 6	Data[1, 2] Logical Span ID (MSB, LSB)
7	Data[3] Channel

0x16 ISDN Secondary D Channel

Byte	AIB Field Description
2	Address Element: 0x16
3	Data Length: 0x07
4	Data[0] Slot Number
5, 6	Data[1, 2] Primary Logical Span ID (MSB, LSB)
7	Data[3] Primary D Channel
8,9	Data[4, 5] Secondary Logical Span ID (MSB, LSB)
10	Data[6] Secondary D Channel

0x17 Subrate

Byte	AIB Field Description
2	Address Element: 0x17
3	Data Length: 0x04
4, 5	Data[0, 1] Logical Span ID (MSB, LSB)
6	Data[2] Channel
7	Data[3] Subrate ID 0x00 Bits 6,7 (16Kbs, 4:1 TDM Mode) 0x01 Bits 5,4 (16Kbs, 4:1 TDM Mode) 0x02 Bits 3,2 (16Kbs, 4:1 TDM Mode) 0x03 Bits 1,0 (16Kbs, 4:1 TDM Mode)

0x1A EXNET Ring

To address a logical ring ID, use the EXNET Ring Address Element

Byte	AIB Field Description
2	Address Element: 0x1A
3	Data Length: 0x01
4	Data[0] Logical Ring ID

0x1C SS7 Destination ID

Byte	AIB Field Description
2	Address Element: 0x1C
3	Data Length: 0x03
4	Data[0] Stack ID
5, 6	Data[1, 2] Destination ID

0x1D SS7 Route

Byte	AIB Field Description
2	Address Element: 0x1D
3	Data Length: 0x03
4	Data[0] Stack ID
5, 6	Data[1, 2] Route ID

0x1E SS7 Link Set

Byte	AIB Field Description
2	Address Element: 0x1E
3	Data Length: 0x02
4	Data[0] Stack ID Stack assigned by the host 0x00 - 0x03
5	Data[1] Link Set ID The signaling link set to assign this link to: 0x00-0x23 Signaling Link Sets 0-35 0xFF Deconfigure the specified signaling link

0x1F SS7 Link Set/Link

Byte	AIB Field Description
2	Address Element: 0x1F
3	Data Length: 0x03
4	Data[0] Stack ID (0-3)
5	Data[1] Link Set ID Indicates the signaling link set to which this link is being assigned. 0x00-0x23 Signaling Link Sets 0-35
6	Data[2] Link ID 0x00-0x0F - Primary SS7 Card 0x10-0x1F - Secondary SS7 Card

0x20 SS7 Destination/Route

Byte	AIB Field Description	
2	Address Element: 0x20	
3	Data Length:	0x05
4	Data[0]	Stack ID
5, 6	Data[1, 2]	Destination ID 0x0000 - 0x007F
7, 8	Data[3, 4]	Route ID 0x0000 - 0x01FF

0x21 SS7 Slot

Byte	AIB Field Description	
2	Address Element: 0x21	
3	Data Length:	0x02
4	Data[0]	Slot Number
5	Data[1]	Stack ID

0x22 DSP Chip

This AIB is used in the *DSP SIMM Configure* (0x00C0) message and *Service State Configure* message to specify an individual DSP chip.

Byte	AIB Field Description	
2	Address Element: 0x22	
3	Data Length:	0x03
4	Data[0]	Slot Number
5	Data[1]	SIMM Number: 0x00 Module 0 0x01 Module 1 0xFF (all DSP chips on all Modules)
6	Data[2]	DSP Chip Number: 0x00 DSP 0 0x01 DSP 1 0x02 DSP 2 0x03 DSP 3 0xFF (all DSPs on this Module)

0x23 Slot/Subrate

Byte	AIB Field Description	
2	Address Element:	0x23
3	Data Length:	0x05
4	Data[0]	Slot Number
5, 6	Data[1, 2]	Logical Span ID (MSB, LSB)
7	Data[3]	Channel
8	Data[4]	Subrate ID 0x00 Bits 6,7 (16Kbs, 4:1 TDM Mode) 0x01 Bits 5,4 (16Kbs, 4:1 TDM Mode) 0x02 Bits 3,2 (16Kbs, 4:1 TDM Mode) 0x03 Bits 1,0 (16Kbs, 4:1 TDM Mode)

0x25 Subrate Channel

Byte	AIB Field Description	
2	Address Method	0x00 (Individual AEs)
3	Number of AEs	0x01
4	Address Element	0x25
5	Data Length	0x05
6, 7	Data[0, 1]	Logical Span ID (MSB, LSB)
8, 9	Data[2, 3]	Subrate Channel
10	Data[4]	Number of bits in Subrate Channel

0x27 ISDN BRI TEI

Byte	AIB Field Description	
2	Address Element:	0x27
3	Data Length:	0x07
4, 5	Data[0, 1]	Logical Span ID (MSB, LSB)
6	Data[2]	Channel
7	Data[3]	Subrate ID 0x00 Bits 6,7 (16Kbs, 4:1 TDM Mode) 0x01 Bits 5,4 (16Kbs, 4:1 TDM Mode) 0x02 Bits 3,2 (16Kbs, 4:1 TDM Mode) 0x03 Bits 1,0 (16Kbs, 4:1 TDM Mode)
8	Data[4]	SAPI
9	Data[5]	CES
10	Data[6]	TEI

0x29 Router

In the *Route Control* message, to use the internal router to find a destination channel, use the Router address type. Currently, the Router Handle is always 0xFFFE.

Byte	AIB Field Description	
2	Address Element:	0x29
3	Data Length:	0x02
4, 5	Data[0, 1]	Router Handle

0x2A SS7 Subsystem Number

Byte	AIB Field Description	
2	Address Element:	0x2A
3	Data Length:	0x03
4	Data[0]	Stack ID
5	Data[1]	SCCP Subsystem Number
6	Data[2]	0x00 (Reserved)

0x2C V5 ID

If this AIB is used, the user port ID must be specified in a V5 Subscriber ID ICB (0x28). In the *PPL Event Indication* message this AIB and the Channel (0x0D) AIB are used for the V5 Layer 3P PSTN protocol. The V5 Configure and V5 Configuration Query messages also use this AIB.

Byte	AIB Field Description	
2	Address Element:	0x2C
3	Data Length	0x02
4	Data[0]	V5 ID (MSB)
5	Data[1]	V5 ID (LSB)

0x2D V5 ID, Link ID

The V5 ID, Link ID address type is used for Link blocking and unblocking.

Byte	AIB Field Description	
2	Address Element:	0x2D
3	Data Length	0x03
4	Data[0]	V5 ID (MSB)
5	Data[1]	V5 ID (LSB)
6	Data[2]	Link ID

0x2F V5 ID, User Port

The V5 ID, User Port address type is used for the following PPL Events: User Port Status Request and V5 Subscriber Query.

Byte	AIB Field Description
2	Address Element: 0x2F
3	Data Length 0x04
4	Data[0] V5 ID (MSB)
5	Data[1] V5 ID (LSB)
6	Data[2] User Port (MSB)
7	Data[3] User Port (LSB)

0x30 V5 ID, User Port, Link, Timeslot

Byte	AIB Field Description	
2	Address Element: 0x30	
3	Data Length	0x06
4	Data[0]	V5 ID (MSB)
5	Data[1]	V5 ID (LSB)
6	Data[2]	Link ID
7	Data[3]	Timeslot
8	Data[4]	User Port (MSB)
9	Data[5]	User Port (LSB)

0x31 SS7 Virtual CIC

Byte	AIB Field Description	
2	Address Element: 0x31	
3	Data Length:	0x07
4	Data[0]	Stack ID
5-8	Data[1-4]	DPC (4 Bytes)
9, 10	Data[5, 6]	CIC

0x32 DS3 Offset

Byte	AIB Field Description	
2	Address Element: 0x32	
3	Data Length:	0x02
4	Data[0]	Slot Number
5	Data[1]	Offset
		This value is always 0.

0x36 MCC

The MCC Group and Channel components use the MCC address type. Currently, the MCC is always 0xFFFE.

Byte	AIB Field Description	
2	Address Element: 0x36	
3	Data Length:	0x02
4, 5	Data[0, 1]	Handle

0x3A D Channel Data Link Connection Identifier

Byte	AIB Field Description
2	Address Element: 0x3A
3	Data Length: 0x04
4	Data[0] Slot of ISDN card
5	Data[1] D-Channels Offset (0-31)
6	Data[2] SAPI
7	Data[3] TEI

0x3C Number of CICS

Byte	AIB Field Description
2	Address Element: 0x3C
3	Data Length: 0x04
4, 5	Data[0, 1] Logical Span ID (MSB, LSB)
6	Data[2] Channel
7	Data[3] Number of Channels

0x3D Ring Configuration Query

Use the Ring Configuration Query AIB to query ring configuration.

Important! If you attempt to query the ring configuration while another query is in progress, the *Response Status* will be 0x3F (Ring Configuration Query Already in Progress)

Byte	AIB Field Description
2	Address Element: 0x3D
3	Data Length: 0x01
4	Data[0] Slot Number of EXNET® Card (Enter 0xFF to query all rings in the system)

0x42 VoIP Module

Byte	AIB Field Description	
2	Address Element: 0x42	
3	Data Length:	0x02
4	Data[0]	Slot Number
5	Data[1]	Module

0x45 Child Conference ID

Use this AIB to create a Child Conference using the *Resource Create* message. The Child Conference ID should be set to 0xFFFF. The AIB in the response to the *Resource Create* message will contain the Parent Conference ID as well as the Child Conference ID, which is allocated by the CSP.

Use this AIB to connect to a Child Conference with the *Resource Connect* message. Indicate the actual Child Conference ID.

Byte	AIB Field Description	
2	Address Element: 0x45	
3	Data Length:	0x04
4, 5	Data[0]	Parent Conference ID
6, 7	Data[1]	Child Conference ID

0x48 Ring Communication Link

This AIB enables synchronizing of an isolated node with a server node in an SS7/EXS multi-node system. If a node gets isolated for a period of time, any changes occurring on that node while it was in isolation get synchronized with the server node. This AIB takes out-of-service (OOS) the ring communication (RCOMM) link between the SS7 in the server node (that controls the remote CICs) and the remote node. An in-service (INS) message will re-establish the link. Messages should be sent to the SS7 server node only. Entry from both ACTIVE and STANBY matrix and SS7 are removed using only one *Service State Configure*. This AIB is only used in *Service State Configure*.

Important! Since this is a brute force mechanism to re-initialize the link, it should be used only rarely. SS7 application tasks are not informed about this link de-configuration so they behave as though the node is still accessible. This AIB should be used only when

there are no calls and no configuration. It should also be used in a pair of OOS and INS.

Byte	AIB Field Description
2	Address Element: 0x48
3	Data Length: 0x02
4	Stack ID
5	Logical Node ID of remote node

0x52 IP Signaling Series 3 Card ID or Expanded Logical Node ID

This AIB is generic. It can be used in any of the newer (introduced in 8.0 and above) Configure and Query messages for any component. If you use this AIB in the *IP Signaling Series 3 Card Configure* message (0x0100) you must also use AIB 0x53.

Byte	AIB Field Description
2	Address Element: 0x52
3	Data Length: 0x02
4	Data[0, 1] Expanded Logical Node ID This field addresses the node/IP Signaling Series 3 Card ID and is used by the host application to route the messages to the proper node/ IP Signaling Series 3 Card. Logical Node ID in the header is ignored (it is set to 0xFE) and Expanded Logical Node ID is used to address the node.

0x53 Object Type

This AIB is generic. It can be used in any Configure and Query message introduced in Release 8.0 and above to configure and query any component. The Object Type field indicates the type of object to be configured, and the Object Instance ID field indicates the ID of the particular instance of the Object in the Object Type field.

Byte	AIB Field Description
2	Address Element: 0x53
3	Data Length: 0x04

Byte	AIB Field Description
4	Data[0, 1] Object Type This field indicates the type of object for which the configuration is meant. For example, to configure an SS7 Series 3 Card, the type is SS7 Stack Type. 0x0001 V.52 Interface 0x0002 SS7 Stack 0x0003 ISDN 0x0004 H323 0x0005 SINAP SS7 0x0020 IP Signaling Series 3 Card Interface 0xFFFF All object types
5, 6	Data[2, 3] Object Instance ID (CSP Matrix Series 3 Card ID) This field addresses the particular instance of the Object listed in the Object Type field. 0xFFFF means that the Object Instance ID does not matter.

0x55 Conference ID

Use this AE with the Slot AE (unless noted) in the following messages:

Play File Start

Play File Stop

Play File Modify

Record File Start

Record File Stop

Record File Modify

System Configuration Query (no Slot AE)

Generic Report (no Slot AE)

Byte	AIB Field Description
2	Address Element: 0x55 Conference ID
3	Data Length: 0x02
4, 5	Data[0, 1]: Conference ID (MSB, LSB)

0x64 Global Object Type

The Global Object Type Address Element defines an object type such as a channel, span, or link. The object type list is common to all objects so no two objects in the system should have the same object type.

Byte	Address Element Field Description	
0	Address Element:	0x64
1	Data Length:	0x02
2, 3	Data[0,1]	0x1000 - SS7 Signaling 0x1001 - SS7 Stack 0x1020 - M3UA 0x1021 - Stack Parameter 0x1022 - Local Application Server 0x1023 - Remote Application Server 0x1024 - Local Application Server Process 0x1025 - Remote Application Server Process 0x1026 - Local Signaling Gateway 0x1027 - Remote Signaling Gateway 0x1028 - Local Signaling Gateway Process 0x1029 - Remote Signaling Gateway Process 0x102A - Route Set 0x102B - Connection

0x65 Global Object ID

The Object ID Address Element always comes after an Object Type and defines a specific instance of the Object Type. Each object has to give the range of valid IDs it can support.

Byte	AIB Field Description	
0	Address Element:	0x65
1	Data Length:	0x02
2,3	Data[0,1]	Object ID Number

0x67 Functional Area

Defines the part of the system affected by the message.

Byte	Address Element Field Description	
2	Address Element	0x67
3	Length	0x02
4,5	Data	0x0001 - Signaling

0x73 IP Signaling Series 3 Card ID

Byte	AIB Field Description
2	Address Element: 0x73
3	Data Length: 0x02
4	Data[0, 1] IP Signaling Series 3 Card ID

0x7F SIP Stack ID

Byte	AIB Field Description
2	Address Element: 0x7F
3	Data Length: 0x04
4	Reserved: 0xFF
5	Stack ID: 0x00
6, 7	Data [0,1] Call Handle. Tied to the Call ID. Used to address a particular SIP dialog. 0xFF 0xFF when querying.

3 Information Control Blocks

Overview

Purpose This chapter lists the Information Control Blocks (ICBs) used in the EXS API for the CSP. The ICBs are listed by type (Action, Data, and Extended Data). The Action ICBs are listed first, then the Data ICBs, and finally the Extended Data ICBs. Within each of these three sections, the ICBs are listed in hexadecimal numerical order.

Introduction to ICBs

Generic ICB Format The table below shows the generic format for an ICB.

ICB Type	
ICB ID	
Data Length	
Data[0]	
:	

ICB Type

0x01 - Action

0x02 - Data

0x03 - Extended

ICB Subtype

The subtype defines the specific action the ICB performs, or the data it contains. The subtype values are defined in the ICB Subtype section of this chapter. In each API message that uses ICBs, the subtypes supported for that message are listed.

Important! Some Action ICB subtype values are used to indicate different ICBs, but they are unique within a particular message. For example, in the *Inseize Control* message, subtype 0x01 is the Report Call Processing Event ICB, while in the *Outseize Control* message, subtype 0x01 is the Scan For Wink N ICB.

Also, a specific ICB may have a different subtype value in different messages. For example, in the *Inseize Control* message, the Report Call Processing Event ICB is subtype 0x01, while in the *Outseize Control* message it is 0x04.

Make sure you use the proper ICB subtype value as defined in the specific API message. Data ICB subtypes are unique system-wide.

ICB Data Length

The length of the data to follow.

ICB Data

The data for each subtype is defined in the ICB Data section of this chapter, and in any API message which supports the subtype. Not all ICBs include data.

In each message, the data for each supported ICB is presented in the following format, if applicable:

ICB Data Length
ICB Data [0]
:

Action ICBs

Most hex IDs are used to represent more than one Action ICB. Two or more ICBs with the same number are never used in the same API message. ICBs marked with an asterisk (*) contain no data, so their data length is zero (0x00) and there is no table in this section.

Hex	Action ICB Name	Used in These Messages
0x00	Null*	<i>Inseize Control, Route Control, Outseize Control</i>
0x01	Free System Resources	<i>Connect With Data</i>
	Report Call Processing Event	<i>Inseize Control</i>
	Scan For Wink N	<i>Route Control, Outseize Control</i>
0x02	Release Mode Configure	<i>Connect With Data</i>
	Generate Call Processing Event	<i>Inseize Control</i>
	Scan For ANI Request Off-hook*	<i>Route Control, Outseize Control</i>
0x03	Receive Stage N Address Data	<i>Inseize Control</i>
	Scan For Dial Tone*	<i>Route Control, Outseize Control</i>
0x04	Wait For Host Control*	<i>Inseize Control</i>
	Report Call Processing Event	<i>Route Control, Outseize Control</i>
0x05	Report Incoming Call*	<i>Inseize Control</i>
	Outpulse Stage N Address Data	<i>Route Control, Outseize Control</i>
0x06	Report Incoming Call With Address Digits*	<i>Inseize Control</i>
	Wait For Host Address Data*	<i>Route Control, Outseize Control</i>
0x07	Generate Inseizure Acknowledgment*	<i>Inseize Control</i>
	Wait For Host Control*	<i>Route Control, Outseize Control</i>
	Query Load Version	<i>TFTP Manage</i>
0x08	Send Host Acknowledgment*	<i>Inseize Control, Route Control, Outseize Control</i>
	Query Load Filename	<i>TFTP Manage</i>
0x09	Use Instruction List	<i>Inseize Control</i>
	Do Call Progress Analysis	<i>Route Control, Outseize Control</i>
0x0A	Delay N Milliseconds	<i>Inseize Control</i>
	Seize*	<i>Route Control, Outseize Control</i>
0x0B	Use Instruction List	<i>Route Control, Outseize Control</i>
0x0D	Cancel R2 Receiver*	<i>Route Control, Outseize Control</i>
0x0E	Scan For Backward R2 Signal*	<i>Route Control, Outseize Control</i>
0x0F	Wait For Host Control with Answer Supervision*	<i>Route Control, Outseize Control</i>
0x10	Do CPA Without Line Signaling*	<i>Route Control, Outseize Control</i>
0x11	Delay N Milliseconds	<i>Route Control, Outseize Control</i>
0x33	Enable Multi-Host*	<i>Multi-Host Control</i>
0x34	Disable Multi-Host*	<i>Multi-Host Control</i>
0x35	Host Query Message Registration	<i>Multi-Host Control</i>
0x36	Connections Query*	<i>Multi-Host Control</i>

0x37	Local CSP Matrix Series 3 Card Configuration Filename	<i>Multi-Host Control</i>
0x38	Query Hard-registered PPL Components	<i>Multi-Host Control</i>
0x39	Query Soft-registered PPL Components	<i>Multi-Host Control</i>

0x01 Free System Resources

Used in:

Connect with Data message

ICB Type	0x01 (Action)
ICB ID	0x01
Data Length	0x02
Data[0]	Channels to Free Resources For 0x01 Channel A 0x02 Channel B 0x03 Both Channels
Data[1]	Resources to Cancel or Free 0x01 Digit Receiver 0x02 Transmit Call Progress Tone 0x03 Both Resources

0x01 Scan For Wink N

Used in:

Route Control message

Outseize Control message

ICB Type	0x01 (Action)
ICB ID	0x01 Scan for Wink N
Data Length	0x01
Data[0]	Wink Number (0x01–0x08)

0x01 Report Call Processing Event

Used in:

Inseize Control message

ICB Type	0x01 (Action)										
ICB ID	0x01 or 0x04 Report Call Processing Event										
Data Length	0x02 (for Digits event) 0x01 (for all other events) .										
Data[0]	<p>Call Processing Event</p> <ul style="list-style-type: none"> 0x00 No Event 0x01 Off-hook 0x02 Digits 0x03 Wink 1 0x04 Wink 2 0x05 Wink 3 0x06 Wink 4 0x07 Wink 5 0x08 Wink 6 0x09 Wink 7 0x0A Wink 8 0x0B Dial Tone 0x0C Reserved <p>All other events are reserved and must be 0x00.</p>										
Data[1]	<p>Stage Numbers</p> <p>This field applies only when the Call Processing Event is Digits (0x02). This field is a bit mask. You can report multiple stages at the same time. Valid entries for the bits of this field range are as follows:</p> <table border="0"> <thead> <tr> <th style="text-align: left;"><u>Bit</u></th> <th></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Stage 1</td> </tr> <tr> <td>1</td> <td>Stage 2</td> </tr> <tr> <td>2</td> <td>Stage 3</td> </tr> <tr> <td>3</td> <td>Stage 4</td> </tr> </tbody> </table> <p>Bits 4-7 are reserved and must be 0x00.</p>	<u>Bit</u>		0	Stage 1	1	Stage 2	2	Stage 3	3	Stage 4
<u>Bit</u>											
0	Stage 1										
1	Stage 2										
2	Stage 3										
3	Stage 4										

0x02 Release Mode Configure

Used in:

Connect with Data message

ICB Count	Variable
ICB Type	0x01 (Action)
ICB ID	0x02
Data Length	0x02
Data[0]	Channel A Release Mode 0x01 Park Channel A if B releases 0x02 Release Channel A if B releases
Data[1]	Channel B Release Mode 0x01 Park Channel B if A releases 0x02 Release Channel B if A releases

0x02 Generate Call Processing Event

Used in:

Inseize Control message

ICB Count	Variable
ICB Type	0x01 (Action)
ICB ID	0x02
Data Length	0x02 (for R2 Signaling Events) 0x01 (for all other events)
Data[0]	Call Processing Event NOTE: Entries marked with an asterisk (*) require a Backward R2 Signaling Value in Data[1]. 0x01 ANI Request Off-hook 0x02 Reserved 0x03 Wink 1 0x04 Wink 2 0x05 Wink 3 0x06 Wink 4 0x07 Wink 5 0x08 Wink 6 0x09 Wink 7 0x0A Wink 8 0x0B* Backward Pulsed R2 Signal 0x0C* Backward Compelled R2 Signal 0x0D* Backward R2 Signal with Completion Event 0x0F* Backward Compelled or Pulsed R2 Signal
Data[1]	Backward R2 Signaling Value Entries above marked with an asterisk use values 0x01-0x0F. All others use 0x00.

0x03 Receive Stage N Address Data

Used in:

Inseize Control message

ICB Type	0x01 (Action)
ICB ID	0x03
Data Length	0x01
Data[0]	Stage Number (0x01–0x04)

0x04 Report Call Processing Event

Used in:

Route Control message*Outseize Control* message*Inseize Control* message

ICB Type	0x01 (Action)
ICB ID	0x04
Data Length	0x02 (for Digits event) 0x01 (for all other events)
Data[0]	<p>Call Processing Event</p> <ul style="list-style-type: none"> 0x00 No Event 0x01 Off-hook 0x02 Digits 0x03 Wink 1 0x04 Wink 2 0x05 Wink 3 0x06 Wink 4 0x07 Wink 5 0x08 Wink 6 0x09 Wink 7 0x0A Wink 8 0x0B Dial Tone 0x0C Reserved 0x0D Backward R2 Signal 0x0E First Digit
Data[1]	<p>Stage Numbers</p> <p>This field applies only when the Call Processing Event is Digits (0x02). This field is a bit mask. You can report multiple stages at the same time. Valid entries for the bits of this field range are as follows:</p> <p><u>Bit</u></p> <ul style="list-style-type: none"> 0 Stage 1 1 Stage 2 2 Stage 3 3 Stage 4

0x05 Output Stage N Address Data

Used in:

Route Control message*Outseize Control* message

ICB Type	0x01 (Action)
ICB ID	0x05
Data Length	0x01
Data[0]	Stage Number (0x01–0x14)

0x07 Query Load Version

Used in:

TFTP Manage message

ICB Type	0x01
ICB ID	0x07 Query Load Version
Data Length	0x01
Data[0]	Load Number

0x08 Send Host Acknowledgment

Used in:

Route Control message*Outseize Control* message*Inseize Control* message

ICB Type	0x01 (Action)
ICB ID	0x08
Data Length	0x00

0x08 Query Load Filename

Used in:

TFTP Manage message

ICB Type	0x01
ICB ID	0x08 Query Load Filename
Data Length	0x01
Data[0]	Load Number

0x09 Do Call Progress Analysis

Used in:

Route Control message*Outseize Control* message

ICB Type	0x01 (Action)
ICB ID	0x09
Data Length	0x01
Data[0]	Class 0x00 North American Default 0x01 Dialtone 0x02 CPC Detection 0x03 Energy Detection

0x09 Use Instruction List

0x09 Used in:

Inseize Control message

ICB Type	0x01 (Action)
ICB ID	0x09
Data Length	0x01
Data[0]	Instruction Number (0x01–0x14)

0x0A Delay N Milliseconds

See also 0x11, Delay N Milliseconds as used in the *Route Control* and *Outseize* messages

Delay values are measured in units of 10 milliseconds. The maximum delay value is 10 seconds (0x03E8).

Used in:

Inseize Control message

ICB Type	0x01 (Action)
ICB ID	0x0A or 0x11
Data Length	0x02
Data[0, 1]	Delay Value (MSB, LSB)

0x0B Use Instruction List

Used in:

Route Control message*Outseize Control* message

ICB Type	0x01 (Action)
ICB ID	0x0B
Data Length	0x01
Data[0]	Instruction Number (0x01–0x14)

0x10 Do Call Progress Analysis Without Line Signaling

Used in:

Route Control message*Outseize Control* message

ICB Type	0x01 (Action)
ICB ID	0x10
Data Length	0x01
Data[0]	Class 0x00 North American Default 0x01 Dialtone 0x02 CPC Detection 0x03 Energy Detection

0x11 Delay N Milliseconds

Delay values are measured in units of 10 milliseconds. The maximum delay value is 10 seconds (0x03E8).

Used in:

Route Control message*Outseize Control* message

ICB Type	0x01 (Action)
ICB ID	0x11
Data Length	0x02
Data[0]	Delay Value (2 Bytes)

0x35 Query Host Message Registration

This query will return a list of API messages which the specified host is registered to receive (see Response Data).

Used in:

Multi-Host Configure message

ICB Type	0x01 (Action)
ICB ID	0x35
Data Length	0x04
Data[0-3]	Host ID

0x36 Query Host Connections

Used in:

Multi-Host Configure message

ICB Type	0x01 (Action)
ICB ID	0x36
Data Length	Variable
Data[0-3]	Default Host ID (4 bytes) (00 00 00 00 if not connected)
Data[4-5]	Number of hosts connected (2 bytes)
Data[6-9]	Host ID 1 (4 bytes. First connected additional host)
Data[10-13]	Host ID 2 (4 bytes)
:	:
Data[:]	Host ID N (4 bytes)

0x37 Query Multi-Host Status

Used in:

Multi-Host Configure message

ICB Type	0x01 (Action)
ICB ID	0x37
Data Length	Variable
Data[0-3]	Default Host ID (4 bytes)
Data[4]	Status 0x00 - Disabled 0x01 - Enabled

0x38 Query Hard-registered PPL Components

This query will return a list of PPL components from which the specified host is registered to receive *PPL Event Indication* messages (see Response Data).

Used in:

Multi-Host Configure message

ICB Type	0x01 (Action)
ICB ID	0x38
Data Length	0x04
Data[0–3]	Host ID

0x39 Query Soft-registered PPL Components

This query will return a list of PPL components from which the specified host is registered to receive *PPL Event Indication* messages (see Response Data).

Used in:

Multi-Host Configure message

ICB Type	0x01 (Action)
ICB ID	0x39
Data Length	0x04
Data[0–3]	Host ID

Data ICBs

Hex	Data ICB Name	Used in These Messages
0x01	Stage N Address Data (Host-Supplied Digits)	<i>Route Control, Outseize Control</i>
0x02	Stage N Address Data (Previously-Inputted Digits)	<i>Route Control, Outseize Control</i>
0x10	ISDN Formatted IEs	<i>Channel Release Request, Release Channel with Data, Request For Service With Data, Route Control, Outseize Control, PPL Event Indication, PPL Event Request</i>
0x11	ISDN Raw IEs	<i>Channel Release Request, Release Channel with Data, Request For Service With Data, Route Control, Outseize Control, PPL Event Indication, PPL Event Request</i>
0x12	SS7 Parameters	<i>Channel Release Request, Channel Released With Data, Connect With Data, Release Channel with Data, Route Control, Outseize Control, PPL Event Indication, PPL Event Request, Request For Service With Data</i>
0x14	PPL Argument 2	<i>PPL Event Request, PPL Event Indication</i>
0x15	DASS2/DPNSS Raw Data	<i>Route Control, Outseize Control, Release Channel with Data, PPL Event Indication</i>
0x16	SS7 Protocol Violation	<i>PPL Event Indication, PPL Event Request</i>
0x17	ISDN Formatted IEs with Event	<i>Connect With Data</i>
0x18	ISDN Raw IEs with Event	<i>Connect With Data</i>
0x1A	Call Type	<i>Outseize Control, Route Control</i>
0x1C	SS7 TUP Formatted Fields	<i>Route Control, Outseize Control, PPL Event Request, Release Channel with Data, Request For Service With Data, PPL Event Indication, Connect With Data</i>
0x1E	Generic PPL ICB	<i>PPL Event Indication, PPL Event Request, Route Control, Channel Release with Data</i>
0x1F	SS7 Unformatted Raw Parameters	<i>PPL Event Indication, Channel Release with Data, Release Channel with Data</i>
0x20	SS7 SCCP Parameters	<i>PPL Event Indication, PPL Event Request</i>
0x21	SS7 TCAP Parameters	<i>PPL Event Indication, PPL Event Request</i>
0x22	Raw SS7 Data Parameters	
0x23	BT IUP Parameters	<i>Channel Release with Data, Release Channel with Data, Request For Service With Data</i>
0x24	Product License	<i>Product License Download</i>
0x25	ISDN Segmented	<i>Outseize Control, Request For Service With Data, Release Channel with Data, Channel Release with Data, PPL Event Indication, PPL Event Request, Route Control</i>
0x37	Local CSP Matrix Series 3 Card Configuration Filename	<i>TFTP Manage</i>

Hex	Data ICB Name	Used in These Messages
0x38	Local CSP Matrix Series 3 Card Server IP Address	<i>TFTP Manage</i>
0x39	Adjacent CSP Matrix Series 3 Card Configuration Filename	<i>TFTP Manage</i>
0x3A	Adjacent CSP Matrix Series 3 Card Server IP Address	<i>TFTP Manage</i>
0x3B	Timestamp	<i>TFTP Manage</i>
0x3C	Save Options	<i>TFTP Manage</i>
0x3D	Load Version	<i>TFTP Manage</i>
0x3E	Load Filename	<i>TFTP Manage</i>
0x47	Enable CSP-Initiated Messages	<i>Multi-Host Configure</i>
0x48	Disable CSP-Initiated Messages	<i>Multi-Host Configure</i>
0x49	Enable <i>PPL Event Indication</i> messages	<i>Multi-Host Configure</i>
0x4B	Deregister CSP-Initiated Messages	<i>Multi-Host Configure</i>
0x4C	Soft Register PPL Component IDs	<i>Multi-Host Configure</i>
0x50	Hard Register All CSP Initiated Messages	<i>Multi-Host Configure</i>
0x51	Soft Register All CSP Initiated Messages	<i>Multi-Host Configure</i>
0x52	Deregister All CSP Initiated Messages	<i>Multi-Host Configure</i>
0x53	Hard Register All PPL Component ID	<i>Multi-Host Configure</i>
0x54	Soft Register All PPL Component IDs	<i>Multi-Host Configure</i>
0x55	Deregister All PPL Component IDs	<i>Multi-Host Configure</i>
0x4A	Disable <i>PPL Event Indication</i> messages	<i>Multi-Host Configure</i>
0x5B	IP Signaling Series 3 Card ID	<i>PPL Event Indication</i>
0x5C	H.323 Hookflash Received (no data)	<i>PPL Event Indication</i>
0x5D	H.323 Signal Input Received	<i>PPL Event Indication</i>
0x5E	H.323 Signal Update Input Received	<i>PPL Event Indication</i>
0x5F	H.323 Alphanumeric Input Received	<i>PPL Event Indication</i>
0x62	Remote Endpoints UII Capabilities	<i>PPL Event Indication</i>
0x65	TCAP Primitive	<i>PPL Event Indication, PPL Event Request</i>
0x66	SS7 Address Information	<i>Request for Service with Data, Outseize Control</i>
0xFF	PPL General Purpose Register Data	<i>PPL Event Indication, PPL Event Request</i>

0x01 Stage N Address Data

(Host-Supplied Digits)

Stage N Address Data ICBs use either host-supplied outpulsing or previously inpulsed digits. The format of the ICB depends on the type of digits you use.

The format of the Stage N Address Data ICB using host-supplied digits is as follows:

Used in:

Route Control message*Outseize Control* message

ICB Type	0x02 (Data)
ICB ID	0x01
Data Length	Variable
Data[0]	Stage Number (1–4)
Data[1]	0x01 (Host-Supplied Digits)
Data[2]	Outpulsing Signal Type 0x01 DTMF 0x02 MFR1 (host does not include KP or ST) 0x03 MFR2 0x04 MFR1 (host includes KP and any ST signal) 0x05 Dial Pulse
Data[3]	String Count (1 or 2)
Data[4]	String 1: BCD Digit Count
Data[5]	String 1: 1st BCD Digit Pair
:	:
:	String 1: Last BCD Digit Pair
:	String 2: BCD Digit Count
:	String 2: 1st BCD Digit Pair
:	:
:	String 2: Last BCD Digit Pair

0x02 Stage N Address Data

(Previously-Inputted Digits)

The format of the Stage N Address Data ICB using previously inputted digits is as follows:

Used in:

Route Control message*Outseize Control* message

ICB Type	0x02 (Data)
ICB ID	0x02
Data Length	0x07
Data[0]	Stage Number (1–4)
Data[1]	0x02 (Previously Inputted Digits)
Data[2]	Outputting Signal Type 0x01 DTMF 0x02 MF1 (host does not include KP or ST) 0x03 MFR2 0x04 MFR1 (host includes KP and any ST signal) 0x06 Dial Pulse 0x06 - Compelled KP (MF-MDR1) (host does not include KP or STI)
Data[3, 4]	Logical Span ID (2 Bytes) (0–63)
Data[5]	Channel
Data[6]	Stage Number (1–4)

0x10 ISDN Formatted IEs

Used in:

Channel Release Request, Channel Released With Data, Release Channel with Data, Request For Service With Data, Route Control, Outseize Control, PPL Event Indication, PPL Event Request messages

ICB Type	0x02 (Data)
ICB ID	0x10
Data Length	Variable
Data[0]	Number of IEs (0–30)

Data[1]	IE 1 Type 0x01 Call Reference 0x02 Called Party Number 0x03 Calling Party Number 0x04 Called Party Subaddress 0x05 Calling Party Subaddress 0x06 Redirecting Number 0x07 Cause 0x08 Progress 0x09 User to User 0x0A Codeset 6 0x0B Codeset 7 0x0C Reserved 0x0D Reserved 0x0E Reserved 0x0F Reserved 0x10 Reserved 0x11 Codeset 5 0x12 Vari-A-Bill
Data[2]	IE 1 Length (0–130)
Data[3]	IE 1 Data[0]
Data[:]	IE n Type
Data[:]	IE n Length (0–130)
Data[:]	IE n Data[0]

0x11 ISDN Raw IEs

Any Raw IE can be inserted by the host for proprietary applications. Raw IEs will be inserted after any Formatted IEs (consult Q.931). The Q.931 recommendation specifies the format of the Raw IEs. IEs are always validated before issuing them in an ISDN message. The ISDN protocols ensure that the mandatory IEs are included with each ISDN message. See *ITU-T Q.931* for a list of valid Raw IEs.

Used in:

Channel Release Request, Channel Released With Data, Release Channel with Data, Request For Service With Data, Route Control, Outseize Control, PPL Event Indication, PPL Event Request messages

ICB Type	0x02 (Data)
ICB ID	0x11
Data Length	Variable
Data[0]	Number of IEs (0–30)
Data[1]	IE Type (see ITU-T Q.931)
Data[2]	IE Length (0–130)
Data[3]	IE Data[0]

0x12 SS7 Parameters

Used in:

Channel Release Request, Channel Released With Data, Connect With Data, Release Channel with Data, Route Control, Outseize Control, PPL Event Indication, PPL Event Request messages.

ICB Type	0x02 (Data)
ICB ID	0x12
Data Length	Variable
ISUP Message ID	See the list of supported ANSI ISUP message types, in the <i>API Developer's Guide: Common Channel Signaling</i> .
Data[0]	Number of Parameters
Data[1]	Parameter 1 Type
Data[2]	Parameter 1 Length
Data[3]	Parameter 1 Data[0] (consult ANSI or ITU-TS recommendations for data)
Data[:]	Parameter n Type
Data[:]	Parameter n Length
Data[:]	Parameter n Data[0] (consult ANSI or ITU-TS recommendations for data)

0x14 PPL Argument 2 Data

Used in:

PPL Event Request message

ICB Type	0x12 (Data)
ICB ID	0x14
Data Length	0x01
Data[0]	State Number (0-255)

0x15 DASS2/DPNSS Raw Data

The DASS2/DPNSS Raw Data ICB is a Raw DASS2/DPNSS frame that starts at the message/group octet (it does not include the address and control octets).

Used in:

Route Control message

Outseize Control message

ICB Type	0x02 (Data)
ICB ID	0x15
Data Length	Variable
Data[0]	Raw DASS2/DPNSS Message/Group Octet
Data[1]	Remaining DASS2/DPNSS Frame

0x16 SS7 Protocol Violation Data

Used in:

PPL Event Indication message*PPL Event Request* message

ICB Type	0x02 (Data)
ICB ID	0x16
Data Length	0x05
Data[0, 1]	PPL Component ID
Data[2]	<p>Error Type</p> <p>NOTE: Entries marked with a dagger (†) report the ISUP message type only and do not report an ISUP parameter.</p> <p>0x01† Unrecognized Message Type</p> <p>0x02 Unrecognized Parameter ID</p> <p>0x03 Minimum Parameter Length Violation</p> <p>0x04 Maximum Parameter Length Violation</p> <p>0x05† Corrupt Parameter Data</p> <p>0x06† Mandatory Parameters Missing</p> <p>0x07 Range Field of Range/Status Parameter is zero</p>
Data[3]	<p>ISUP Message ID</p> <p>To see a list of supported ANSI ISUP message types, refer to the SS7 Chapter in the API Developer's Guide: Common Channel Signaling</p>
Data[4]	<p>ISUP Parameter ID</p> <p>See the API Developer's Guide: CCS. If the value of the Error Type field is marked with a †, the value of this field is 0x00.</p>

0x17 ISDN Formatted IEs With Event

Used in:

Connect with Data message

ICB Type	0x02 (Data)
ICB ID	0x17
Data Length	Variable
Data[0, 1]	Event
Data[2]	Number of IEs (0–30)

Data[3]	IE Type 0x01 Call Reference 0x02 Called Party Number 0x03 Calling Party Number 0x04 Called Party Subaddress 0x05 Calling Party Subaddress 0x06 Redirecting Number 0x07 Cause 0x08 Progress Indicator 0x09 User-to-User 0x0A Codeset 6 0x0B Codeset 7 0x0C Reserved 0x0D Reserved 0x0E Reserved 0x0F Reserved 0x10 Reserved 0x11 Codeset 5 0x12 Vari-A-Bill
Data[4]	IE Length (0–130)
Data[5]	IE Data[0]

0x18 ISDN Raw IEs with Event

You can insert any valid IE into this ICB. The ISDN messages listed below have the listed IEs automatically inserted unless overridden by the host. All other messages have no IEs automatically inserted.

SETUP

- Call Reference (Formatted IE)
- Bearer Capability (Raw IE)
- Network-specific IE (Raw IE) (Unless B channel's Network Type is set to 1, see *B Channel Configure*)
- Channel ID (will always be inserted for Primary Rate, cannot be inserted by host)

RELEASE

- Cause (Formatted IE)

Used in:

Connect with Data message

ICB Type	0x02 (Data)
ICB ID	0x18
Data Length	Variable
Data[0, 1]	Event
Data[2]	Number of IEs (0–30)

Data[3]	IE Type (see ITU-T Q.931)
Data[4]	IE Length (0–130)
Data[5]	IE Data[0]

0x1A Call Type

Call Types:

0x0001L3P Controlled

0x0002 Host Controlled

If the value of the Call Type field is 0x0001, the remaining data fields Data[2] through Data[n], represent the digits of the called party number.

The called party digits are noted in ASCII/IA5 format. For example, the digit “5” is indicated as 0x35.

If the value of the Call Type field is 0x0002, data fields Data[2] through Data[n] are not included in the ICB.

Used in:

Route Control message

Outseize Control message

ICB Type	0x02 (Data)
ICB ID	0x1A
Data Length	Variable
Data[0, 1]	Call Type MSB, LSB
Data[2]	Called Party Digit
Data[3]	Called Party Digit
Data[4]	Called Party Digit

0x1C SS7 TUP Formatted Fields

Used in:

Channel Release Request, Channel Released With Data, Connect With Data, Release Channel with Data, Route Control, Outseize Control, PPL Event Indication, PPL Event Request messages.

ICB Type	0x02 (Data)
ICB ID	0x1C
Data Length	Variable
Data[0, 1]	TUP Message Group/Type
Data[2]	Number of Fields (variable)
Data[3]	Field 1 ID (Refer to the <i>TUP Message Format Configure</i> message)
Data[4]	Field 1 Data Length (variable)
Data[5]	Field 1 Data[0]

Data[:]	Field n ID (Refer to the <i>TUP Message Format Configure</i> message)
Data[:]	Field n Data Length (variable)
Data[:]	Field n Data[0]

0x1E Generic PPL

This ICB is used in various scenarios including call routing and establishing IP calls.

Used in:

Route Control message

PPL Event Indication message

PPL Event Request message

Channel Release with Data message

ICB Type	0x02 (Data)
ICB ID	0x1E
Data Length	Total number of bytes below this line
Data[0, 1]	Total Number of TLVs
Data[2, 3]	TLV 1 Tag
Data[4, 5]	TLV 1 Length
Data[6]	TLV 1Data[0]
Data[:]	:
Data[:]	TLV 1Data[n]
Data[2, 3]	TLV <i>n</i> Tag
Data[4, 5]	TLV <i>n</i> Length
Data[6]	TLV <i>n</i> Data[0]

Data[:]	:
Data[:]	TLV n Data[n]
TLVs	0x0008 Criteria Type 0x0009 Terminating Channel 0x000F Router Protocol ID 0x0013 Routing Method 0x0065 Physical VoIP Channel 0x0071 Virtual VoIP Channel 0x0100 RTP Payload Type (VDAC-ONE) 0x0100 RTP Payload Type (IPN Series 2) 0x0102 RTP Silence Suppression 0x0103 RTP Echo Cancellation 0x0116 Channel Service TLV 0x0117 Alerting Propagation Mode 0x0118 Companding Conversion Mode 0x0119 Media Offer Stage 0x011A Call Agent Mode Physical Channel ID 0x01C2 Minimum Jitter Buffer Delay 0x01C3 Maximum Jitter Buffer Delay (IPN-2 Card) 0x01C4 Adaptation Rate 0x01C5 Fax Type Enable 0x01C7 Fax/Modem Bypass Coder Type 0x01C8 Protocol Type 0x01D1 RTP Payload Redundancy 0x01D2 Fax Payload Redundancy 0x01D4 Type of Service 0x01DF UDP Source Port Validate 0x01E1 Fax Compatibility Mode 0x01E2 RFC 2833 Enable 0x01E6 Source T.38 Port 0x01E7 Destination T.38 Port 0x01E8 Source RTCP Port 0x01E9 Destination RTCP Port 0x01EB RTP Timer Timeout (VDAC-ONE) 0x01EC Media Inactivity Detection Timer 0x11 Signaling Route Test Control Status 0x12 Signaling Route Test Control DPC

0x1F SS7 Unformatted Raw Parameters

This ICB is used to send the unrecognized message/message received on unequipped CIC from SPRC. The data bytes contain the entire ISUP message.

Used in:

PPL Event Indication message

ICB Type	0x02 (Data)
ICB ID	0x1F
Data Length	Variable
Data[0-3]	DPC (OPC of the message originator)

Data[4-5]	CIC
Data[6]	ISUP Message ID See the Appendix in the <i>API Developer's Guide: Common Channel Signaling</i> . For a full listing, see ITU Q.763 or ANSI T1.113.
Data[n]	SS7 Raw Data for message including pointers. (Structure depends on SS7 message received.)

0x20 SS7 SCCP Parameters

Used in:

PPL Event Indication message

PPL Event Request message

ICB Type	0x02 (Data)
ICB ID	0x20
Data Length	Variable
Data[0]	Raw Parameter Data: Parameter Length Data
Data[n]	:

0x21 SS7 TCAP Parameters

Used in:

PPL Event Indication message

PPL Event Request message

ICB Type	0x02 (Data)
ICB ID	0x21
ICB Length	Variable
TCAP Dialogue ID (4 Bytes)	
Parameter Type (2 Bytes)	
Parameter Length (2 Bytes)	
Parameter Data	

0x22 Raw SS7 Data Parameters

This ICB is a general purpose one used to indicate different data formats. For example, it is used in the PPL event indications of CQS (Circuit Query Sending) to indicate local and remote status. Data definitions for this can be found in the *API Developer's Guide: Common Channel Signaling*. This ICB is also used by CVT (Circuit Validation Test) procedure for PPL event indications from component

CVS (Circuit Validation Sending.). This ICB is also included in the ISUP CPC (0x0012) PPL event indication 0x4E. For more data information on this refer to the *Developer's Guide: Common Channel Signaling, PPL Events*.

ICB Type	0x02 (Data)
ICB ID	0x22
Data Length	Variable
Data[0]	Raw Parameter Data: Parameter Length Data
:	:
Data[n]	:

0x23 BT IUP Parameters

The BT IUP Parameters ICB (0x23) is used to transport any BT IUP Parameters message between the host and the CSP.

Important! In general, messages will contain one BT IUP ICB with one or more BT IUP TLVs.

ICB Type	0x02 (Data)
ICB ID	0x23
Data Length	Variable
Data[0]	Tag value corresponding to a unique element of a BT IUP message.
Data[1]	Number of bytes to follow (usually 0x02)
Data[2]	H0 (See following table.)
Data[3]	H1 (See following table.)
Data[2+]	Message Data

Message Types

The following table lists the messages in alphabetical order. The table also includes the H0 and H1 octets in binary code and the decimal code in brackets.

Message Title	Message Group Code (H0)	Message Type Code (H1)
Additional Call Information (ACI) - Type 1	00000111[7]	00000100[4]
Additional Call Information (ACI) - Type 2	00000111[7]	00000100[4]
Additional Call Information (ACI) - Type 3	00000111[7]	00000100[4]
Additional Call Information (ACI) - Type 4	00000111[7]	00000100[4]

Additional Call Information (ACI) - Type 5	00000111[7]	00000100[4]
Additional Call Information (ACI) - Type 6	00000111[7]	00000100[4]
Additional Call Information (ACI) - Type 7	00000111[7]	00000100[4]
Additional Setup Information (ASUI) - Type 1	00000001[1]	00000100[4]
Additional Setup Information (ASUI) - Type 2	00000001[1]	00000100[4]
Address Complete Message (ACM)	00000011[3]	00000000[0]
Answer (ANS)	00000100[4]	00000000[0]
Blocking (BLO)	00000101[5]	00000001[1]
Blocking Acknowledgement (BLA)	00000101[5]	00000011[3]
Call Drop Back (CDB)	00000011[3]	00001010[10]
Circuit Free (CCF)	00000101[5]	00000000[0]
Clear (CLR)	00000100[4]	00000001[1]
Coin and Fee Checking (CFC)	00000100[4]	00000100[4]
Confusion (CFN)	00000111[7]	00000000[0]
Congestion (CNG)	00000011[3]	00000010[2]
Connection Not Admitted (CNA)	00000011[3]	00000100[4]
Enveloped ISUP Message (EIM)	00001000[8]	00000010[2]
Enveloped ISUP Segmented Message (EISM)	00001000[8]	10000010[130]
Extend Call Message (ECM)	00000100[4]	00000111[7]
Final Address Message (FAM)	00000000[0]	00000011[3]
Howler (HLR)	00000100[4]	00000110[6]
Initial Address Message (IAM)	00000000[0]	00000000[0]
Initial and Final Address Message (IFAM)	00000000[0]	00000001[1]
ISDN Composite Service Information Message (SIM) -Type 1	00000111[7]	00000001[1]
ISDN Composite Service Information Message (SIM) -Type 2	00000111[7]	00000001[1]
ISDN Composite Service Information Message (SIM) -Type 3	00000111[7]	00000001[1]
ISDN Composite Service Information Message (SIM) -Type 4	00000111[7]	00000001[1]
ISDN Composite Service Information Message (SIM) -Type 5	00000111[7]	00000001[1]
ISDN Composite Service Information Message (SIM) -Type 6	00000111[7]	00000001[1]
ISDN Composite Service Information Message (SIM) -Type 7	00000111[7]	00000001[1]
ISDN Composite Service Information Message (SIM) -Type 8	00000111[7]	00000001[1]
Nodal End-to-End Data (NEED)	00000111[7]	00001011[11]
Operator Condition Message (OCM)	00000111[7]	00000101[5]
Operator Override (OOR)	00000100[4]	00000101[5]
Overload (OLM)	00000101[5]	00000101[5]
Protocol Negotiation Message (PNM)	00001000[8]	00000001[1]
Re-answer (RAN)	00000100[4]	00000010[2]
Release (REL)	00000100[4]	00000011[3]
Repeat Attempt Message (RAM)	00000011[3]	00000101[5]
Send Additional Set-up Information (SASUI)	00000010[2]	00000100[4]
Send All Digits (SAD)	00000010[2]	00000011[3]
Send N Digits (SND)	00000010[2]	00000010[2]
Send Service Message (SSM)	00000111[7]	00000010[2]
Service (SER)	00000111[7]	00000011[3]
Subscriber Engaged Message (SEM)	00000011[3]	00000110[6]
Subscriber Out-of-Order (SOO)	00000011[3]	00000111[7]

Subscriber Address Message (SAM)	00000000[0]	0000010[2]
Swap	00000111[7]	00000111[7]
Terminal Congestion Message (TCM)	00000011[3]	00000011[7]
Unblocking (UBL)	00000101[5]	00000010[2]
Unblocking Acknowledgement (UBA)	00000101[5]	00000010[2]
User-initiated Suspend (SUS)	00000111[7]	00001100[12]
User-initiated Resume (RES)	00000111[7]	00001101[13]
User to User Data (UUD)	00000111[7]	00000110[6]

0x24 Product License

A Product License is 16 bytes long. Bytes 0 and 1 are the Key Type and describe the type of license that is being administered. Bytes 2-15 are encrypted data.

Used in:

Product License Download message (0x0079).

ICB Type	0x02 (Data)
ICB ID	0x24

Data Length	0x10
Data[0, 1]	<p>Key Type</p> <p>The key type associates the software key with a particular locked module or hardware upgrade.</p> <p>NOTE: For SS7 User Parts (except SCCP/TCAP) only one key type is required for each instance of a User Part.</p> <p>0x3031 Enable SS7 TUP on all stacks</p> <p>0x3032 Enable SS7 ISUP on all stacks</p> <p>0x3033 Enable SS7 IUP on all stacks</p> <p>0x3034 Enable SS7 SCCP/TCAP on an active SS7 card</p> <p>0x3036 Enables ability to configure one or more SS7 Virtual CICs (This key type was for previous releases. For the current release, use key type 0x3037 below.)</p> <p>0x3037 Enables ability to configure one or more SS7 Virtual Span/Channels</p> <p>0x3038 Enables ability to configure SIP</p> <p>0x3039 Enables ability to configure Call Agent with IP Virtual Span/Channels</p> <p>0x3041 Enables/Disables RFC 2833 (SIP)</p> <p>0x3042 Enables ability to configure V.52</p> <p>0x3043 Enables ability to configure ISDN LAPD</p> <p>0x3044 Enables ability to configure QSIG</p> <p>0x3045 Resource Points</p> <p>0x3130 Enable System Software</p> <p>0x3131 Enable Channels on IP Network Interface Series 2 Card</p> <p>0x3132 M3UA Software</p> <p>0x3134 Enable MTP3 to Host</p> <p>0x3830 Enable n-links on an SS7 card</p> <p>0x3831 Enable n-spans on a T-ONE card</p> <p>0x3832 Enable n-spans on an E-ONE card</p> <p>0x3833 Enable n-spans on a J-ONE card</p>
Data[2–15]	<p>Encrypted Data</p> <p>A 14-byte alphanumeric string containing encrypted data</p>

0x25 ISDN Segmented Message

Used in:

Outseize Control message

Request For Service With Data message

Release Channel With Data message

Channel Release Request message

PPL Event Indication message

PPL Event Request message

Route Control message

ICB Type	0x02
ICB ID	0x25 ISDN Segmented Message

Data Length	0x02
Data[0]	Number of segments remaining
Data[1]	Segmented ISDN message type

0x26 Channel Pad Value

Used in:

Connect with Data message

ICB Type	0x02
ICB ID	0x26 ISDN Segmented Message
Data Length	0x02
Data[0]	Channel A Pad Value The dB gain/loss adjustment of the signal transmitted to Channel A 0x00 +3 dB Gain/Loss 0x01 0 dB 0x02 2 dB 0x03 3 dB 0x04 4 dB 0x05 6 dB 0x06 9 dB
Data[1]	Channel B Pad Value The dB gain/loss adjustment of the signal transmitted to Channel B 0x00 +3 dB Gain/Loss 0x01 0 dB 0x02 2 dB 0x03 3 dB 0x04 4 dB 0x05 6 dB 0x06 9 dB

0x37 Local CSP Matrix Series 3 Card Configuration Filename

Used in:

TFTP Manage API message

ICB Type	0x02
ICB ID	0x37 Local CSP Matrix Series 3 Card Configuration Filename
Data Length	Variable
Data[0]	First Character
:	:
:	Null Termination

0x38 Local Matrix Server IP Address

Used in:

TFTP Manage API message

ICB Type	0x02
ICB ID	0x38 Local Matrix Server IP Address
Data Length	0x04
Data[0-3]	Segment

0x39 Adjacent CSP Matrix Series 3 Card Configuration Filename

Used in:

TFTP Manage API message

ICB Type	0x02
ICB ID	0x39 Adjacent CSP Matrix Series 3 Card Configuration Filename
Data Length	Variable
Data[0]	First Character
:	:
:	Null Termination

0x3A Adjacent Matrix Server IP Address

Used in:

TFTP Manage API message

ICB Type	0x02
ICB ID	0x3A Adjacent Matrix Server IP Address
Data Length	0x04
Data[0]	Most Significant Segment
:	:
:	Least Significant Segment

0x3B Timestamp

Used in:

TFTP Manage API message

ICB Type	0x02
ICB ID	0x3B Timestamp
Data Length	0x04
Data[0-3]	Seconds since January 1, 1970

0x3C Save Options

Used in:

TFTP Manage API message

ICB Type	0x02
ICB ID	0x3C Save Options
Data Length	Variable
Data[0]	First Load with Save Option Set
:	:
:	Last Load with Save Option Set

0x3D Load Version

Used in:

TFTP Manage API message

ICB Type	0x02
ICB ID	0x3D Load Version
Data Length	0x04
Data[0]	Load Number
Data[1]	Major Revision
Data[2]	Minor Revision
Data[3]	Build Number

0x3E Load Filename

Used in:

TFTP Manage API message

ICB Type	0x02
ICB ID	0x3E Load Filename
Data Length	Variable
Data[0]	Load Number
Data[1]	First Character
:	:
:	Null Termination

0x47 Enable Switch-Initiated Messages

Used in:

Multi-Host Configure message

ICB Type	0x02 (Data)
ICB ID	0x47
Data Length	Variable
Data[0-3]	Host ID

Data[4+]	API Message 1 Type (Two bytes)
:	:
:	API Message n Type

0x48 Disable Switch-Initiated Messages

Used in:

Multi-Host Configure message

ICB Type	0x02 (Data)
ICB ID	0x48
Data Length	Variable
Data[0-3]	Host ID
Data[4+]	API Message 1 Type (2 bytes)
:	:
:	API Message n Type

0x49 Enable PPL Event Indication Messages

Used in:

Multi-Host Configure message

ICB Type	0x02 (Data)
ICB ID	0x49
Data Length	Variable
Data[0-3]	Host ID
Data[4+]	PPL Component ID 1 (2 bytes)
:	:
:	PPL Component ID n (2 bytes)

0x4A Disable PPL Event Indication Messages

Used in:

Multi-Host Configure message

ICB Type	0x02 (Data)
ICB ID	0x4A
Data Length	Variable
Data[0-3]	Host ID
Data[4+]	PPL Component ID 1 (2 bytes)
:	:
:	PPL Component ID n (2 bytes)

0x4B Deregister Switch-Initiated Messages

Used in:

Multi-Host Configure message

ICB Type	0x02 (Data)
ICB ID	0x4B
Data Length	Variable
Data[0-3]	Host ID
Data [4+]	API Message 1 Type (2 bytes)
:	:
:	API Message n Type

0x4C Deregister PPL Components

Used in:

Multi-Host Configure message

ICB Type	0x02 (Data)
ICB ID	0x4C
Data Length	Variable
Data[0-3]	Host ID
Data [4+]	PPL Component ID 1 (2 bytes)
:	:
:	PPL Component ID n (2 bytes)

0x50 Hard Register All Switch-Initiated Messages

Used in:

Multi-Host Configure message

ICB Type	0x02 (Data)
ICB ID	0x50
Data Length	0x04
Data[0-3]	Host ID

0x51 Soft Register All Switch-Initiated Messages

Used in:

Multi-Host Configure message

ICB Type	0x02 (Data)
ICB ID	0x51
Data Length	0x04
Data[0-3]	Host ID

0x52 Deregister All Switch-Initiated Messages

Used in:

Multi-Host Configure message

ICB Type	0x02 (Data)
ICB ID	0x52
Data Length	0x04
Data[0-3]	Host ID

0x53 Hard Register All PPL Component IDs

Used in:

Multi-Host Configure message

ICB Type	0x02 (Data)
ICB ID	0x53
Data Length	0x04
Data[0-3]	Host ID

0x54 Soft Register All PPL Component IDs

Used in:

Multi-Host Configure message

ICB Type	0x02 (Data)
ICB ID	0x54
Data Length	0x04
Data[0-3]	Host ID

0x55 Deregister All PPL Component IDs

Used in:

Multi-Host Configure message

ICB Type	0x02 (Data)
ICB ID	0x55
Data Length	0x04
Data[0-3]	Host ID

0x5B IP Signaling Series 3 Card ID

Used in:

PPL Event Indication message

ICB Type	0x02 (Data)
----------	-------------

ICB ID	0x5B
Data Length	0x02
Data[0-1]	IP Signaling Series 3 Card ID (2 bytes)

0x5D H.323 DTMF Signal Input Received

A DTMF digit received from the PSTN or IP side of a call.

Used in:

PPL Event Indication message

ICB Type	0x02 (Data)
ICB ID	0x5D
Data Length	0x04
Data[0-1]	Digits
Data[2-3]	Duration (ms)

0x5E H.323 Signal Update Input Received

A DTMF digit update received from PSTN or IP side of the call.

Used in:

PPL Event Indication message

ICB Type	0x02 (Data)
ICB ID	0x5E
Data Length	0x04
Data[0-1]	Digits
Data[2-3]	Duration (ms)

0x5F H.323 Alphanumeric Input Received

An alphanumeric digit received from IP side of the call.

Used in:

PPL Event Indication message

ICB Type	0x02 (Data)
ICB ID	0x5F
Data Length	0x04
Data[0-1]	Digits
Data[1-2]	0x0000 (no duration)

0x62 Remote Endpoints UII Capabilities

Used in:

PPL Event Indication message

ICB Type	0x02 (Data)
ICB ID	0x62
Data Length	0x02
Data[0-1]	0x0000 - Does not support UII 0x0005 - Supports UII

0x65 TCAP Primitive

Used in:

PPL Event Indication message*PPL Event Request* message

ICB Type	0x02 (Data)
ICB ID	0x65
Data Length	0x0nn
Data[0-3]	TCAP Dialogue ID
Data[4-5]	TCAP TC-Primitive ID
Data[6+]	TCAP TC-Parameter TLVs
:	:

0x66 SS7 Address Information

Used in:

Request for Service with Data message*Outseize Control* message

ICB Type	0x02 (Data)
ICB ID	0x66
Data Length	0x0C
Data[0-3]	CIC (4 Bytes)
Data[4-7]	OPC (4 Bytes)
Data[8-11]	DPC (4 Bytes)

0xFF PPL General Purpose Register Data

Used in:

PPL Event Indication message*PPL Event Request* message

ICB Type	0x02 (Data)
ICB ID	0xFF
Data Length	Variable
Data[0-7]	Event Data

Extended Data ICBs

0x0012 SS7 Formatted Parameters

The SS7 Formatted Parameters Extended Data ICB supports Data greater than 255 bytes for call processing with ISUP messages.

For an SS7 channel in ANSI ISUP CRM/CRA, the *Outseize Control* message must include an SS7 Parameters ICB. This allows a CRM message type to be part of the SS7 Parameter ICB that is sent in the *Outseize Control* message. To send an IAM message after an *Outseize Control* message with CRM, a separate *PPL Event Request* for IAM must be made.

Used in:

Outseize Control message

Request for Service with Data message

PPL Event Request message

PPL Event Indication message

ICB Type	0x03 (Extended Data)
ICB ID	0x0012
Data Length (2 Bytes)	Variable
ISUP Message ID	Consult ITU Q.764
Data[0]	Number of Parameters
Data[1]	Parameter Type
Data[2]	Parameter Length
Data[3]	Parameter Data[0] (Consult Q.764 for details)

0x001E Generic PPL

Used in:

Route Control message

Connect With Data message

ICB Type	0x03 (Extended Data)
ICB ID	0x001E
Data Length (2 bytes)	Variable
Number of TLVs (2bytes)	Variable
Tag	TLV 1 Tag
Length	TLV 1 Length
Value	TLV 1 Value
:	TLV n Tag

:	TLV n Length
:	TLV n Value
TLVs	0x11 Signaling Route Test Control Status 0x12 Signaling Route Test Control DPC 0x0008 Criteria Type 0x000F Router Protocol ID 0x0013 Routing Method 0x0065 Physical VoIP Channel 0x0071 Virtual VoIP Channel 0x0100 RTP Payload Type (VDAC-ONE) 0x0100 RTP Payload Type (IPN Series 2) 0x0102 RTP Silence Suppression 0x0103 RTP Echo Cancellation 0x0116 Channel Service TLV 0x0117 Alerting Propagation Mode 0x0118 Companding Conversion Mode 0x011A Call Agent Mode Physical Channel ID 0x0141 Modify Connection 0x01C2 Minimum Jitter Buffer Delay 0x01C3 Maximum Jitter Buffer Delay (IPN-2 Card) 0x01C4 Adaptation Rate 0x01C7 Fax/Modem Bypass Coder Type 0x0612 Connection Type

0x0020 SS7 SCCP Parameters

Used in:

PPL Event Indication message

PPL Event Request message

ICB Type	0x03 (Extended)
ICB ID	0x0020
Data Length (2 Bytes)	Variable
Data[0]	Raw Parameter Data: Parameter Length Data
Data[n]	:

0x0021 SS7 TCAP Parameters

Used in:

PPL Event Indication message

PPL Event Request message

ICB Type	0x03 (Extended)
ICB ID	0x0021
ICB Length (2 Bytes)	Variable
TCAP Dialogue ID (4 Bytes)	

Parameter Type (2 Bytes)	
Parameter Length (2 Bytes)	
Parameter Data	

0x0026 Channel Pad Value

Used in:

Connect with Data message

ICB Type	0x02
ICB ID	0x26 ISDN Segmented Message
Data Length	0x02
Data[0]	<p>Channel A Pad Value</p> <p>The dB gain/loss adjustment of the signal transmitted to Channel A</p> <p>0x00 +3 dB Gain/Loss 0x01 0 dB 0x02 2 dB 0x03 3 dB 0x04 4 dB 0x05 6 dB 0x06 9 dB</p>
Data[1]	<p>Channel B Pad Value</p> <p>The dB gain/loss adjustment of the signal transmitted to Channel B</p> <p>0x00 +3 dB Gain/Loss 0x01 0 dB 0x02 2 dB 0x03 3 dB 0x04 4 dB 0x05 6 dB 0x06 9 dB</p>

0x0028 V5 Subscriber ID

This ICB is used for call processing messages to and from Layer 3.

Used in:

Channel Released with Data message

Request for Service with Address Data message

PPL Event Indication message

Route Control message

ICB Type	0x03 (Extended Data)
ICB ID	0x0028
Data[1, 2]	
Data Length	(0x00, 0x04), (2 Bytes)
Data[3, 4]	
Data[5, 6]	V5 ID (MSB, LSB), (2 Bytes)
Data[7, 8]	V5 User Port ID (MSB, LSB), (2 Bytes)

0x0029 V5 Formatted IEs

Used in:

Channel Released with Data message

Request for Service with Address Data message

Route Control message

PPL Event Indication message

ICB Type	0x03 (Extended Data)
ICB ID	0x0029
Data[1, 2]	
Data Length	(MSB, LSB), (2 Bytes)
Data[3, 4]	
Data[5]	Number of IEs
Data[6]	V5 Formatted IE Type
Data[7]	V5 Formatted IE Length
Data[8]	V5 Formatted IE Data
:	:
Data[N-2]	V5 Formatted IE Type
Data[N-1]	V5 Formatted IE Length
Data[N]	V5 Formatted IE Data

0x002B MTP3 User Part Parameter

Used in:

PPL Event Request message

PPL Event Indication message

ICB Type	0x03 (Extended Data)
ICB Subtype	0x002B MTP3 User Part Parameter
ICB Length	Length of parameters, in bytes
Data[0]	Parameter 1 Type ID
Data[1]	Length of parameter 1 data in bytes

Data[2]	Parameter 1 Value Variable size
:	
Data[:]	Parameter n Type ID
Data[:]	Length of parameter n data in bytes
Data[:]	Parameter n Value Variable size

PPL Parameters

ICB Type	0x03 (Extended Data)
ICB ID	0x0424
Data Length	0x000D
2 bytes	
Data[0-3]	DPC - Destination Point Code
Data[4-7]	OPC - Originating Point Code
Data[8-9]	CIC - Circuit ID Code
Data[10]	SI - Service Indicator
Data[11]	MP - Message Priority
Data[12]	NI - Network Indicator

ICB Type	0x03 (Extended Data)
ICB ID	0x0425
Data Length	0x000C
2 bytes	
Data[0-3]	DPC - Destination Point Code
Data[4-7]	OPC - Originating Point Code
Data[8]	SI - Service Indicator
Data[9]	MP - Message Priority
Data[10]	NI - Network Indicator
Data[11]	SLS - Signaling Link Set

ICB Type	0x03 (Extended Data)
ICB ID	0x0426

Data Length	Up to 0x010E
2 bytes	
Data[0-1]	Length - Length of MSU Data
Data[4-n]	MSU Data - Length up to 0x010C

ICB Type	0x03 (Extended Data)
ICB ID	0x0427
Data Length	0x0006
2 bytes	

Data[0-3]	Affected DPC
Data[4]	<p>Cause Value</p> <p>Value 0 - congestion level 0</p> <p>Value 1 - congestion level 1</p> <p>Value 2 - congestion level 2</p> <p>Value 3 - congestion level 3</p> <p>Value 4 - User Part Unavailable</p> <p>Value 5 - User Part Unequipped</p> <p>Value 6 - User Part Inaccessible</p> <p>Value 7 - Signaling Link Congestion level 0</p> <p>Value 8 - Signaling Link Congestion level 1</p> <p>Value 9: Signaling Link Congestion level 2</p> <p>Value 10: Signaling Link Congestion level 3</p> <p>Value 11: Signaling Link Congestion level 4</p>
Data[5]	User Information Octet value

0x002C V5 User Port Range

This ICB is used for call processing messages to and from Layer 3.

Used in:

PPL Event Indication message

PPL Event Request message

ICB Type	0x03 (Extended Data)
ICB ID Data[1, 2]	0x002C
Data Length Data[3, 4]	(0x00, 0x04), (2 Bytes)
Data[5, 6]	User Ports From (MSB, LSB), (2 Bytes)
Data[7, 8]	User Port s To (MSB, LSB), (2 Bytes)

0x002D Q.752 Parameter

See MTP Parameter IDs from Q.752 for specific Parameter ID information. The format for the Q.752 Parameter is as follows:

Used in:

PPL Event Indication message

ICB Type	0x03 (Extended Data)
ICB ID	0x002D (Q.752 parameters)
ICB Length	(2 Bytes) Length of parameters, in bytes
Parameter 1 Type	(2 Bytes) Parameter Type ID
Parameter 1 Length	(2 Bytes) Length of parameter data in bytes
Parameter 1 Value	Variable size
:	
Parameter n Type	(2 Bytes) Parameter Type ID
Parameter n Length	(2 Bytes) Length of parameter data in bytes
Parameter n Value	Variable

0x002F V5 Statistics Query

This ICB subtype is sent from the host using *PPL Event Request* message for Envelope Layer, LAPV, PSTN, BCC, Control Protocol, Protection Protocol, Link Protocol, and System Management Statistics. The Statistics return to the host in a *PPL Event Indication* message. After receiving V5 Statistics Query from the host, L3P Layer Manager sends a Positive Acknowledgment back to the host, indicating that L3P Layer Manager received the request.

Used in:

PPL Event Request message

PPL Event Indication message

ICB Type	0x03 (Extended Data)
ICB ID Data[1, 2]	0x002F
Data Length Data[3, 4]	(0x00, 0x04), (2 Bytes)
Data[5]	Query Type (0x01, 0x02)
Data[6]	Query Subtype (0x00 - 0x09)
Data[7]	Type (0x00 - 0x07)
Data[8, 9]	C Channel No. or Port No. (MSB, LSB)

Query Types

0x01 V5 Statistics and zero statistics counters

0x02 V5 Statistics and don't zero statistics counters

Query Subtypes

0x00 All Protocol Statistics

0x01 V5.2 ENVELOPE Layer Statistics

0x02 V5.2 LAPV Statistics

0x03 V5.2 PSTN Statistics

0x04 V5.2 CONTROL Protocol Statistics

0x05 V5.2 BCC Protocol Statistics

0x06 V5.2 LINK Protocol Statistics

0x07 V5.2 PROTECTION Protocol Data Link Statistics

0x08 V5.2 PROTECTION Protocol Interface Statistics

0x09 V5.2 SYSTEM MANAGEMENT Statistics

Type

0x00 No Type

The following types are valid for V5.2 LAPV Statistics:

0x01 PSTN Data Link

0x02 BCC Data Link

0x03 CONTROL Data Link

0x04 LINK

0x05 PRIMARY PROTECTION Data Link

0x06 SECONDARY PROTECTION Data Link

C Channel Number

Valid for V5.2 ENVELOPE Layer Statistics and V5.2 PROTECTION Protocol Statistics:

- 0x00 Primary C Channel
- 0x01 Secondary C Channel

Port Number

User port number, and is valid for V5.2 SYSTEM MANAGEMENT Statistics.

0x0030 V5 Statistics Data

Used in:
PPL Event Indication message

ICB Type	0x03 (Extended Data)
ICB ID Data[1, 2]	0x0030
Data Length Data[3, 4]	(MSB, LSB), (2 Bytes)
Data[5]	Data[0]
Data[6]	Data[1]
:	:
Data[N+3]	Data[N]

The following statistics values are sent to the host in the ICB Data:

Data[0] = Query Subtype

Query Subtypes:

- 0x00 All Protocol Statics
- 0x01 V5.2 ENVELOPE Layer Statistics
- 0x02 V5.2 LAPV Statistics
- 0x03 V5.2 PSTN Statistics
- 0x04 V5.2 CONTROL Protocol Statistics
- 0x05 V5.2 BCC Protocol Statistics
- 0x06 V5.2 LINK Protocol Statistics
- 0x07 V5.2 PROTECTION Protocol Data Link Statistics
- 0x08 V5.2 PROTECTION Protocol Interface Statistics
- 0x09 V5.2 SYSTEM MANAGEMENT Statistics

Each data Variable is 4 Bytes long.
 Data[1] means Data[1,2,3,4] bytes

Query Subtype 0: For All Protocols

Data for PSTN Protocol Statistics:

- Data[1] Number of Establish messages transmitted
- Data[5] Number of Establish ack messages transmitted
- Data[9] Number of Signal messages transmitted
- Data[13] Number of Signal acknowledge messages transmitted
- Data[17] Number of Status messages transmitted
- Data[21] Number of Status enquiry messages transmitted
- Data[25] Number of Disconnect messages transmitted
- Data[29] Number of Disconnect complete messages transmitted
- Data[33] Number of Protocol parameter messages transmitted
- Data[37] Number of Establish messages received
- Data[41] Number of Establish ack messages received
- Data[45] Number of Signal messages received
- Data[49] Number of Signal acknowledge messages received
- Data[53] Number of Status messages received
- Data[57] Number of Status enquiry messages received
- Data[61] Number of Disconnect messages received
- Data[65] Number of Disconnect complete messages received
- Data[69] Number of Protocol parameter messages received
- Data[73] Total connections established so far
- Data[77] Number of currently active connections

Data for Control Protocol Statistics:

- Data[81] Port control transmitted
- Data[85] Port control ack transmitted

Data[89] Common control transmitted
Data[93] Common control ack transmitted
Data[97] Port control received
Data[101] Port control ack received
Data[105] Common control received
Data[109] Common control ack received

Data for BCC Protocol Statistics:

Data[113] Allocation messages transmitted
Data[117] Allocation messages received
Data[121] Deallocation requests transmitted
Data[125] Decollating requests received
Data[129] Allocation rejects transmitted
Data[133] Allocation rejects received
Data[137] Allocation complete transmitted
Data[141] Allocation complete received
Data[145] Deallocation complete transmitted
Data[149] Deallocation complete received
Data[153] Audit transmitted
Data[157] Audit received
Data[161] Audit complete transmitted
Data[165] Audit complete received
Data[169] Fault transmitted
Data[173] Fault received
Data[177] Fault ack transmitted
Data[181] Fault ack received

Data[185] Protocol error transmitted

Data[189] Protocol error received

Data for LINK Protocol Statistics:

Data[193] Number of Link Control transmitted

Data[197] Number of Link Control received

Data[201] Number of Link Control Ack transmitted

Data[205] Number of Link Control Ack received

Data for PROTECTION Protocol Data Link:
(Statistics for Primary C Channel)

Data[209] Number of messages transmitted

Data[213] Number of messages received

Data for PROTECTION Protocol Data Link:
(Statistics for Secondary C Channel)

Data[217] Number of messages transmitted

Data[221] Number of messages received

Data for PROTECTION Protocol Interface Statistics:

Data[225] Number of Switchover request transmitted

Data[229] Number of Switchover request received

Data[233] Number of Switchover command transmitted

Data[237] Number of Switchover command received

Data[241] Number of Operator initiated switchover command
transmitted

Data[245] Number of Operator initiated switchover command
received

Data[249] Number of Switchover ack transmitted
Data[253] Number of Switchover ack received
Data[257] Number of Switchover reject transmitted
Data[261] Number of Switchover reject received
Data[265] Number of Reset sequence number command transmitted
Data[269] Number of Reset sequence number command received.
Data[273] Number of Reset sequence number ack transmitted
Data[277] Number of Reset sequence number ack received
Data[281] Number of Protocol error transmitted
Data[285] Number of Protocol error received

Data for SYSTEM MANAGEMENT Statistics:

Data[289] Number of times the ISDN port status T1 timer expired

Query Subtype 1: V5.2 ENVELOPE Layer Statistics:

Data[1] Number of Information frames transmitted
Data[5] Number of Information frames received
Data[9] Number of frames dropped

Query Subtype 2: V5.2 LAPV Statistics:

Data[1] Information frames transmitted
Data[5] Receive ready frames transmitted
Data[9] Receive not ready frames transmitted
Data[13] Reject frames transmitted
Data[17] Set asynchronous balanced mode frames transmitted
Data[21] Unnumbered acknowledge frames transmitted
Data[25] Disconnect mode frames transmitted
Data[29] Information frames transmitted

Data[33] Receive ready frames transmitted
Data[37] Receive not ready frames transmitted
Data[41] Reject frames received
Data[45] Set asynchronous balanced mode frames received
Data[49] Unnumbered acknowledge frames received
Data[53] Disconnect mode frames received
Data[57] Invalid frames received
Data[61] Set asynchronous balanced mode errors

Query Subtype 3: V5.2 PSTN Statistics:

Data[1] Number of Establish messages transmitted
Data[5] Number of Establish ack messages transmitted
Data[9] Number of Signal messages transmitted
Data[13] Number of Signal acknowledge messages transmitted
Data[17] Number of Status messages transmitted
Data[21] Number of Status enquiry messages transmitted
Data[25] Number of Disconnect messages transmitted
Data[29] Number of Disconnect complete messages transmitted
Data[33] Number of Protocol parameter messages transmitted
Data[37] Number of Establish messages received
Data[41] Number of Establish ack messages received
Data[45] Number of Signal messages received
Data[49] Number of Signal acknowledge messages received
Data[53] Number of Status messages received
Data[57] Number of Status enquiry messages received
Data[61] Number of Disconnect messages received
Data[65] Number of Disconnect complete messages received

Data[69] Number of Protocol parameter messages received

Data[73] Total connections established so far

Data[77] Number of currently active connections

Query Subtype 4: V5.2 CONTROL Protocol Statistics:

Data[1] Port control transmitted

Data[5] Port control ack transmitted

Data[9] Common control transmitted

Data[13] Common control ack transmitted

Data[17] Port control received

Data[21] Port control ack received

Data[25] Common control received

Data[29] Common control ack received

Query Subtype 5: V5.2 BCC Protocol Statistics:

Data[1] Allocation messages transmitted

Data[5] Allocation messages received

Data[9] Deallocation requests transmitted

Data[13] Decollating requests received

Data[17] Allocation rejects transmitted

Data[21] Allocation rejects received

Data[25] Allocation complete transmitted.

Data[29] Allocation complete received

Data[33] Deallocation complete transmitted

Data[37] Deallocation complete received

Data[41] Audit transmitted

Data[45] Audit received

Data[49] Audit complete transmitted

Data[53] Audit complete received

Data[57] Fault transmitted

Data[61] Fault received

Data[65] Fault ack transmitted

Data[69] Fault ack received

Data[73] Protocol error transmitted

Data[77] Protocol error received

Query Subtype 6: V5.2 LINK protocol Statistics:

Data[1] Number of Link Control transmitted

Data[5] Number of Link Control received

Data[9] Number of Link Control Ack transmitted

Data[13] Number of Link Control Ack received

Query Subtype 7: V5.2 PROTECTION Protocol Datalink Statistics:

Data[1] Number of messages transmitted

Data[5] Number of messages received

Query Subtype 8: V5.2 PROTECTION Protocol Interface:

Data[1] Number of Switchover request transmitted

Data[5] Number of Switchover request received

Data[9] Number of Switchover command transmitted

Data[13] Number of Switchover command received

Data[17] Number of Operator initiated switchover command transmitted

Data[21] Number of Operator initiated switchover command received

Data[25] Number of Switchover ack transmitted

Data[29] Number of Switchover ack received

- Data[33] Number of Switchover reject transmitted
- Data[37] Number of Switchover reject received
- Data[41] Number of Reset sequence number command transmitted
- Data[45] Number of Reset sequence number command received
- Data[49] Number of Reset sequence number ack transmitted
- Data[53] Number of Reset sequence number ack received
- Data[57] Number of Protocol error transmitted
- Data[61] Number of Protocol error received

Query Subtype 9: V5.2 SYSTEM MANAGEMENT Statistics:

- Data[1] Number of times the ISDN port status T1 timer expired

0x0031 V5 Status Indication

This ICB is sent from the CSP in a *PPL Event Indication* message for specifying the V5 Status Indications.

Used in:

PPL Event Indication message

ICB Type	0x03 (Extended Data)
ICB ID Data[1, 2]	0x0031
Data Length Data[3, 4]	(MSB, LSB), (2 Bytes)
Data[5]	Data[0] Reason

When PPL Config byte 6 of component 0x93 is set to '0' the following Events and Reasons are sent to host:

Event	Reason	Description
2	0	Invalid entity, instance
	1	Configuration already done
	2	Cannot allocate memory
	7	Invalid port type
	8	Ports exceed max ports
	9	Port already present
	10	Invalid element type
	11	Invalid interface number
	131	Link already present
	132	Maximum links already present
	143	Port insert into hash list failed
3	77	Port/all ports deleted
	78	Interface deleted
	79	All links/link deleted
	80	All ports blocked
	81	All links blocked
	144	Accelerated port align OK
5	0	Invalid entity, instance
	11	Invalid interface number
	12	Invalid port ID
	13	Invalid unsolicited action
	74	Invalid trace generation action
	75	Invalid subaction parameter
	76	Invalid element type
	77	Delete port error
	78	Delete interface error
	79	Delete link error
	80	Block port error
	81	Block link error
6	0	Invalid entity and instance
	11	Invalid interface number
	12	Invalid port ID
	134	Invalid port type
	135	Invalid link number
7	0	Invalid entity, instance
	12	Invalid port ID
8	87	Port control protocol: Event out of sequence
	88	Port control protocol: Saved events already reaching limit
	89	Port control protocol: Timer T1 expired twice

10	82	Variant and ID mismatch
	83	V5.1 FSM reporting control protocol DL failure
	84	V5.1 FSM timer expiry
	85	V5.2 interface FSM receiving bad input event
	124	Accelerated port alignment failed
	136	V5.1 FSM getting a bad input event
11	0	Layer 4 unable to bind/unbind to Layer 3
12	86	Invalid message
	90	Message decode error
	99	Invalid Message length
	102	Protocol discriminator error
	103	Layer 3 address error
	104	Invalid message type
	105	Invalid message content
106	Invalid optional IE	
13	90	BCC protocol error msg received
14	91	Layer 4 primitive failed, resource unavailable
15	90	BCC/protection protocol state-event matrix received invalid event
16	98	Layer 4 primitive failed, requested TSAP not found
17	89	Control protocol TCON timer expired
18	82	Startup-Variant ID mismatch
	110	Startup-Layer 2 activation fail
	111	Startup-Link ID failed
	112	Startup complete
	113	Startup failed
19	89	Restart timer expired. Restart failed
	146	Restart complete
22	126	Switchover failed
	127	Switchover complete
	142	Switchover reject received
23	97	Protection protocol message send failure, both DLs down
	108	Message Send failure, invalid Protocol FSM state
24	108	Message received in an invalid protocol FSM state
26	145	Port FSM received an invalid event

28	15	Port unblock indication
	17	Port block indication
	21	Grading information
	50	Unsuccessful activation attempt
	52	Reception of FE2
	62	User failure indication
	63	Network failure indication
30	92	Protection protocol: reset SN error
1	93	Generating protocol error message
1	139	Notification to layer manager of SN misalignment. Layer 3 resets the SN values
1	140	Reset SN indication after action on reset SN Req
1	141	Reset SN acknowledge reception indication
31	92	TSO4 expires
	94	Switchover message not responded to, timer expires
	109	Second expiry of Link Control Protocol Timer
	139	Reset SN acknowledge not received, timer expiry
34	0	Link Dead
	1	Link Up
	3	Remotely blocked
	5	Locally blocked
35	0	C Channel dead
	1	C Channel alive
	2	C Channel blocked
	3	C Channel active
36	0	Interface down
	1	Interface alive
37	0	Port Out of Service (OOS)
	1	Port alive
	2	Port blocked

When PPL Config byte 6 of component 0x93 is set to '1' the following Events and Reasons are sent to host.

Event	Reason	Description
1	3	General Config OK
	4	TSAP Config OK
	5	Interface Config OK
	6	Port Config OK
	133	Link Config OK

2	0	Invalid entity, instance
	1	Configuration already done
	2	Cannot allocate memory
	7	Invalid port type
	8	Ports exceed max ports
	9	Port already present
	10	Invalid element type
	11	Invalid interface number
	131	Link already present
	132	Maximum links already present
	143	Port insert into hash list failed
3	77	Port/all ports deleted
	78	Interface deleted
	79	All links/link deleted
	80	All ports blocked
	81	All links blocked
	144	Accelerated port align OK
4	61	Link operational
	62	Link not operational
	63	Link ID required
	64	Link ID request
	66	Link ID release indication
	67	Link ID reject indication
	68	Link ID failure
	69	Link block indication
	70	Link unblock request
	71	Link unblock indication
	72	Link block request: deferred
	73	Link block request: non-deferred
5	0	Invalid entity, instance
	11	Invalid interface number
	12	Invalid port ID
	13	Invalid unsolicited action
	74	Invalid trace generation action
	75	Invalid subaction parameter
	76	Invalid element type
	77	Delete port error
	78	Delete interface error
	79	Delete link error
	80	Block port error
	81	Block link error

6	0	Invalid entity and instance
	11	Invalid interface number
	12	Invalid port ID
	134	Invalid port type
	135	Invalid link number
7	0	Invalid entity, instance
	12	Invalid port ID
8	87	Port control protocol: Event out of sequence
	88	Port control protocol: Saved events already reaching limit
	89	Port control protocol: Timer T1 expired twice
9	87	Common control protocol: Event out of sequence
	88	Common control protocol: Saved events already reaching limit
	89	Common control protocol: Timer T2 expired twice
10	82	Variant and ID mismatch
	83	V5.1 FSM reporting control protocol DL failure
	84	V5.1 FSM timer expiry
	85	V5.2 interface FSM receiving bad input event
	124	Accelerated port alignment failed
	136	V5.1 FSM getting a bad input event
11	0	Layer 4 unable to bind/unbind to Layer 3
12	86	Invalid message
	90	Message decode error
	99	Invalid Message length
	102	Protocol discriminator error
	103	Layer 3 address error
	104	Invalid message type
	105	Invalid message content
	106	Invalid optional IE
13	90	BCC protocol error msg received
14	91	Layer 4 primitive failed, resource unavailable
15	90	BCC/protection protocol state-event matrix received invalid event
16	98	Layer 4 primitive failed, requested TSAP not found
17	89	Control protocol TCON timer expired

18	82	Startup-Variant ID mismatch
	110	Startup-Layer 2 activation fail
	111	Startup-Link ID failed
	112	Startup complete
	113	Startup failed
19	89	Restart timer expired. Restart failed
	146	Restart complete
20	114	Data link release received, autonomous switchover initiated
	115	Data link release received, autonomous switchover to be initiated on getting control protocol DL release
	116	Data link down indication, standby not configured
	117	BCC DI release, audit of connections requested
21	32	Verify reprovisioning
	33	Switchover to new variant
	34	Variant and ID received
	35	Ready for reprovisioning
	36	Cannot reprovision
	37	Not ready for reprovisioning
	38	Reprovisioning started
22	126	Switchover failed
	127	Switchover complete
	142	Switchover reject received
23	97	Protection protocol message send failure, both DLs down
	108	Message Send failure, invalid protocol FSM state
24	108	Message received in an invalid protocol FSM state
25	128	Layer manager should issue MPH_ID to layer 1
	129	Layer manager should issue MPH_IDR to layer 1
	130	Layer manager should issue MPH_NOR to layer 1
26	145	Port FSM received an invalid event
27	22	ISDN-activate access
	26	ISDN-deactivate access

28	14	Port unblock request
	15	Port unblock indication
	16	Port block request
	17	Port block indication
	21	Grading information
	50	Unsuccessful activation attempt
	51	Reception of FE101
	52	Reception of FE2
	53	DS active
	54	Access activated
	55	Reception of FE105
	56	Access deactivated
	57	Indication of DL failure
	58	ISDN-block D Channel from user port
	59	ISDN-unblock D Channel from user port
	60	Indication of LOS/LFA
	61	Access activation by user
	65	ISDN-PRI AN maintenance
	66	ISDN-PRI AN maintenance
	67	ISDN-PRI AN maintenance
	68	ISDN-PRI AN maintenance
69	ISDN-PRI AN maintenance	
70	ISDN-PRI AN maintenance	
71	ISDN-PRI AN maintenance	
29	100	PSTN Timer T3 expiry
	101	PSTN Timer T4 expiry
30	92	Protection protocol: reset SN error
1	93	Generating protocol error message
1	139	Notification to layer manager of SN misalignment. Layer 3 resets the SN values
1	140	Reset SN indication after action on reset SN Request
1	141	Reset SN acknowledge reception indication
31	92	TSO4 expires
	94	Switchover message not responded to, timer expires
	109	Second expiry of Link Control Protocol Timer
	139	Reset SN acknowledge not received, timer expiry
34	0	Link Dead
	1	Link Up
	3	Remotely blocked
	5	Locally blocked

35	0	C Channel dead
	1	C Channel alive
	2	C Channel blocked
	3	C Channel active
36	0	Interface down
	1	Interface alive
37	0	Port Out of Service (OOS)
	1	Port alive
	2	Port blocked

0x0033 NPDI Universal ICB

The *Route Control* message is sent with the following ICB to obtain a span/channel for an incoming call. This ICB is required for IP and Interworking calls.

The NPDI SIP Response Code TLV (0x2915) appears in this ICB when it is used in the *Channel Released with Data* message and *Release Channel with Data* messages. This ICB is required for the VDAC-ONE and IP Network Interface Series 2 cards.

This ICB is required to support Frequency Shift Keying for the DSP Series 2 card.

The maximum recommended NPDI size is as follows:

- 780 for the *Route Control* message (including the NPDI ICB size)
- 820 bytes for *Outseize Control* message and the other messages below (including the NPDI ICB size)

Used in:

Outseize Control message

PPL Event Indication message

PPL Event Request message

Route Control message

Connect with Data message

Request for Service with Data message

Channel Released with Data message

Release Channel with Data message

ICB Type	0x03 (Extended Data)
ICB ID	0x0033
Data Length	Variable (2 bytes)
Number of TLVs	Variable (2 bytes)
Tag	TLV 1 Tag
Length	TLV 1 Length
Value	TLV 1 Value

:	TLV n Tag
:	TLV n Length
:	TLV n Value
TLVs	<p>TLVs that can be used in this ICB are grouped by the signaling protocol where they can be used:</p> <p>H.323</p> <ul style="list-style-type: none"> 0x02C1 Vendor ID 0x2717 Called Party Number 0x2718 Calling Party Number (Connected Number) 0x2761 Channel Identification 0x27C9 Charge Area Information Parameter 0x2792 Source IP Address 0x2793 Source RTP Port 0x2794 Destination IP Address 0x2795 Destination RTP Port 0x27B0 RTP Payload Type 0x27B1 RTP Payload Size 0x27C1 IP Signaling Series 3 Card ID 0x27C2 Remote End Signaling IP Address (Q.931) 0x27C3 Remote End Signaling Port (Q.931) 0x27C4 Source IP 0x27C5 Source Port 0x27D8 Source H.323 ID 0x27D9 Source URL 0x27DA Source E-mail 0x27DC Remote H.323 ID 0x27DD Remote URL 0x27DE Remote E-mail 0x27E3 Release Cause Code

	<p>Interworking</p> <ul style="list-style-type: none"> 0x2716 Call Identity (Interworking Only) 0x2717 Called Party Number 0x2718 Calling Party Number (Connected Number) 0x271A Circuit State (Interworking Only) 0x271B Cause Indicators (Interworking Only) 0x2725 Forward Call Indicators (Interworking Only) 0x2727 Generic Digit Information (Interworking Only) 0x2729 Generic Number (Interworking Only) 0x272A Generic Reference Number (Interworking Only) 0x272B Hop Counter (Interworking Only) 0x2737 Original Called Number (Interworking Only) 0x2737 Original Called Number (Interworking Only) 0x273D Redirection Information (Interworking Only) 0x273E Redirection Number (Interworking Only) 0x2745 Transit Network Selection (Interworking Only) 0x2749 Bearer Capability (Interworking Only) 0x274E NPDI Message Type 0x274F Progress Indicator (Interworking Only) 0x2761 Channel Identification 0x2750 Display (Interworking Only) 0x2751 High Layer Compatibility (Interworking Only) 0x2752 Low Layer Compatibility (Interworking Only) 0x2753 Calling Party Sub Address (Interworking Only) 0x2754 Called Party Sub Address (Interworking Only) 0x2755 Low Layer Compatibility Layer 1 Info (Interworking Only) 0x2756 Low Layer Compatibility Layer 2 Information (Interworking Only) 0x2757 Low Layer Compatibility Layer 3 Information 0x2758 Pulse Notification (Interworking Only) 0x2759 Call State (Interworking Only) 0x275A Generic Command (Interworking Only) 0x275B Generic Command Response (Interworking Only) 0x275C Signal (Interworking Only) 0x275D Pulsed-Signal (Interworking Only) 0x275E Steady Signal (Interworking Only) 0x275F Recognition Time (Interworking Only) 0x2760 Line Information (Interworking Only) 0x2761 Channel Identification 0x2762 Date and Time (Interworking Only) 0x2763 Carrier Information (Interworking Only) 0x277D Universal Message Envelope (Interworking Only) 0x277E Remote Side Protocol 0x278F Originating Line Information (Interworking Only) 0x27C9 Charge Area Information Parameter 0x27C9 Charge Area Information Parameter 0x278C Connected Number Subaddress (Interworking Only) 0x2792 Source IP Address 0x27C9 Charge Area Information Parameter 0x27CA Carrier Information Parameter (Interworking Only) 0x27CB Non-Presentation Reason (Interworking Only) 0x27D0 Partial Calling Line Identity (Interworking Only)
--	---

SIP

0x2717 Called Party Number
 0x2718 Calling Party Number (Connected Number)
 0x2761 Channel Identification
 0x27C9 Charge Area Information Parameter
 0x2792 Source IP Address
 0x2793 Source RTP Port
 0x2794 Destination IP Address
 0x2795 Destination RTP Port
 0x27AF NPDI SIP Fax Mode
 0x27B0 RTP Payload Type
 0x27B1 RTP Payload Size
 0x27E3 Release Cause Code
 0x290E NPDI SIP Proxy IP Address
 0x290F NPDI SIP Proxy Port
 0x2910 NPDI SIP Route Type
 0x2914 NPDI SIP Local Lookup
 0x2915 NPDI SIP Response Code
 0x2916 SIP A Leg Transport Type
 0x2917 SIP B Leg Transport Type
 0x2919 NPDI SIP To Username
 0x291A NPDI SIP To Password
 0x291B NPDI SIP To Host Name
 0x291C NPDI SIP To Port
 0x291E NPDI SIP Refer to Username
 0x291F NPDI SIP Refer to Password
 0x2920 NPDI SIP Refer to Host Name
 0x2921 NPDI SIP Refer to Port
 0x2923 NPDI SIP From User Name
 0x2924 NPDI SIP From Password
 0x2925 NPDI SIP From Host Name
 0x2926 NPDI SIP From Port
 0x2929 - SIP Referred By Header URI TLV
 0x292E NPDI SIP Contact Password
 0x292F NPDI SIP Contact Hostname
 0x2930 NPDI SIP Contact Port
 0x2931 NPDI SIP Contact Expires
 0x2933 SIP Contact Transport Type
 0x2934 SIP Contact URI User Information Qualifier
 0x2929 - SIP Referred By Header URI TLV
 0x2935 SIP Contact URI Parameters
 0x2936 SIP Tunnel Type
 0x2937 NPDI SIP Authenticate Scheme
 0x2938 NPDI SIP Authentication Realm
 0x2939 NPDI SIP Authenticate Username
 0x293A NPDI SIP Authenticate Password
 0x293B NPDI SIP Authorization Username
 0x293C NPDI SIP Authorization Password
 0x293D NPDI SIP Proxy Authorization Username
 0x293E NPDI SIP Proxy Authorization Password
 0x293F NPDI SIP Authentication Timeout
 0x2941 SIP Authorization Header
 0x2942 SIP Proxy Authorization Header
 0x2943 SIP Authenticate Header
 0x2944 SIP Proxy-Authenticate Header

	SIP (Cont.) 0x294E SIP Remote IP Address 0x294F SIP Remote IP Port 0x2954 SIP Request URI User Name 0x2955 SIP Request URI Host Name 0x2956 SIP Request URI Port 0x2957 SIP Request URI User Information Qualifier 0x2958 SIP Request URI Parameters 0x29D4 SIP Do Not Close Socket 0x29FF Media Local End Point Information 0x2A00 Media Remote End Point Information 0x2A02 Media Per Codec Information 0x2A03 Media Type 0x2A07 Media Port 0x2A08 Media Payload Type 0x2A09 Media Payload Description 0x2A0A Media Payload Size 0x2A0B Media Clock Rate 0x2A0E Media Connection Address 0x2A13 Media Flow Direction Frequency Shift Keying (FSK) 0x0690 Frequency Shift Keying (FSK) Data 0x0691 Terminal Equipment Alerting Signal (TAS) 0x0692 Subscriber Alerting Signal (SAS) 0x0693 FSK Data Transmission Type 0x0693 FSK Data Transmission Type 0x0695 FSK Modem Type
--	--

0x0034 LAPD Frame

Used in:

PPL Event Request message

PPL Event Indications message

ICB Type	0x03 (Extended Data)
ICB ID	0x0034
Data Length (2 bytes)	Length of Layer 3 Information
First Byte of Layer 3 Data	Variable
:	
Last Byte of Layer 3 Data	Variable

0x0035 LAPD Status

Used in:

PPL Event Request message

PPL Event Indications message

ICB Type	0x03 (Extended Data)
ICB ID	0x0035
Data Length	0x0001
LAPD State	Variable

0x0065 TCAP Primitive

Used in:

PPL Event Indication message

PPL Event Request message

ICB Type	0x03 (Extended Data)
ICB ID	0x0065
Data Length	0x0nnn
Data[0-3]	TCAP Dialogue ID
Data[4-5]	TCAP TC-Primitive ID
Data[6+]	TCAP TC-Parameter TLVs
:	:

SS7 and ISDN Values

This section provides data values required for SS7 ICBs, including the following:

- SS7 Parameter IDs
- MTP Statistics Parameter IDs from Q.752

See “SS7 ISUP Message Format Configure 0x6B” for a list of ISUP Message IDs.

SS7 Parameter IDs

The following is a partial list of SS7 parameters. For a complete list and coding, consult the appropriate ANSI or ITU specification.

0x03	Access transport
0x27	Automatic congestion level
0x11	Backward call indicators
0xC6	Business group
0x01	Call reference
0x04	Called party number
0x0A	Calling party number
0x09	Calling party's category
0xC5	Carrier identification
0xEE	Carrier selection information
0x12	Cause indicators
0xEB	Charge number
0x0D	Connection request
0xC3	Egress service
0x07	Forward call indicators
0xC0	Generic address
0xC1	Generic digits
0x0F	Information indicators
0x0E	Information request indicators
0xC4	Jurisdiction
0x06	Nature of connection indicators
0xEF	Network transport
0xE1	Notification indicator
0x29	Optional backward call indicators
0x28	Original called number
0xEA	Originating line information
0x13	Redirection information
0x0B	Redirection number
0xE2	Service activation

0x03	Access transport
0xEC	Service code indicator
0xED	Special processing request
0xE3	Transaction request
0x23	Transit network selection
0x1D	User service information

MTP Parameter IDs from Q.752

Parameter length for each ID listed below is 0x0004 except where noted.

0x0000	<p>MTP3 Timestamp</p> <p>This value is a 1msec resolution, modulus 2^{32} timestamp derived from the internal processing logic of MTP3. The datum of this timestamp is established internally by each MTP3 at the time the signalling stack is configured and cannot be set.</p> <p>As presented, the timestamp should only be used for relative timing of MTP3 Q.752 events and reports within a signaling stack. The timestamp for interval reports identify the time associated with the end of the reporting interval and not the time of the actual report. Consequently, host attempts to correlate the timestamp to a time datum must consider only the 1st interval report of a sequence for this correlation.</p>
0x0001	<p>Interval sequence number</p> <p>This value is a modulus 2^{32} sequence number which identifies the reporting interval. The initial value for the first reporting interval following start or restart of interval reporting is 1.</p>
0x0002	Duration of the link in-service state
0x0003	Duration of the link unavailable state
0x0004	Duration of the link local inhibit state
0x0005	Duration of the link remote inhibit state
0x0006	Duration of the link failed state
0x0007	Duration of the link remote blocked state
0x0008	Count of link failures
0x0009	Count of link restorations
0x000A	Count of link failures due to an abnormal FIBR/BSNR
0x000B	Count of link failures due to delayed ACK
0x000C	Count of link failures due to error rate
0x000D	Count of link failures due to excessive congestion duration
0x000E	Count of link failures due to remote LSSU received
0x000F	Count of link failures due to link test failures
0x0010	Count of link alignment failures
0x0011	Count of link SUs received in error
0x0012	Count of link NACKs received
0x0013	Count of MSUs transmitted
0x0014	Count of octets transmitted
0x0015	Count of link octets retransmitted
0x0016	Count of MSUs received
0x0017	Count of octets received
0x0018	Duration of inaccessibility
0x0019	Count of MSUs discarded during routing

0x001A	Count of MSUs discarded during discrimination
0x001B	Counts of UPU's transmitted for each user part (Parameter Length=0x0040)
0x001C	Counts of UPU's received for each user part (Parameter Length=0x0040)
0x001D	Link failure reason code 0x0001 Abnormal FIBR/BSNR 0x0002 Excessive delayed ACK (T7) 0x0003 Excessive error rate 0x0004 Excessive duration of congestion (T6) 0x0005 LSSU received (SIOS, SIO, SIN, SIE) 0x0006 Changeover MSU received 0x0007 Other
0x001E	User part identifier

ISDN Values

This section provides data values required for ISDN ICBs, including:

- Information Elements
- IE Fields
- Cause Codes

Parameter length for each ID listed below is 0x0004 except where noted.

0x0000	MTP3 Timestamp This value is a 1msec resolution, modulus 2 ³² timestamp derived from the internal processing logic of MTP3. The datum of this timestamp is established internally by each MTP3 at the time the signalling stack is configured and cannot be set. As presented, the timestamp should only be used for relative timing of MTP3 Q.752 events and reports within a signaling stack. The timestamp for interval reports identify the time associated with the end of the reporting interval and not the time of the actual report. Consequently, host attempts to correlate the timestamp to a time datum must consider only the 1 st interval report of a sequence for this correlation.
0x0001	Interval sequence number This value is a modulus 2 ³² sequence number which identifies the reporting interval. The initial value for the first reporting interval following start or restart of interval reporting is 1.
0x0002	Duration of the link in-service state
0x0003	Duration of the link unavailable state
0x0004	Duration of the link local inhibit state
0x0005	Duration of the link remote inhibit state
0x0006	Duration of the link failed state
0x0007	Duration of the link remote blocked state
0x0008	Count of link failures
0x0009	Count of link restorations
0x000A	Count of link failures due to an abnormal FIBR/BSNR
0x000B	Count of link failures due to delayed ACK
0x000C	Count of link failures due to error rate
0x000D	Count of link failures due to excessive congestion duration
0x000E	Count of link failures due to remote LSSU received

0x000F	Count of link failures due to link test failures
0x0010	Count of link alignment failures
0x0011	Count of link SUs received in error
0x0012	Count of link NACKs received
0x0013	Count of MSUs transmitted
0x0014	Count of octets transmitted
0x0015	Count of link octets retransmitted
0x0016	Count of MSUs received
0x0017	Count of octets received
0x0018	Duration of inaccessibility
0x0019	Count of MSUs discarded during routing
0x001A	Count of MSUs discarded during discrimination
0x001B	Counts of UPUs transmitted for each user part (Parameter Length=0x0040)
0x001C	Counts of UPUs received for each user part (Parameter Length=0x0040)
0x001D	Link failure reason code 0x0001 Abnormal FIBR/BSNR 0x0002 Excessive delayed ACK (T7) 0x0003 Excessive error rate 0x0004 Excessive duration of congestion (T6) 0x0005 LSSU received (SIOS, SIO, SIN, SIE) 0x0006 Changeover MSU received 0x0007 Other
0x001E	User part identifier

Information Elements

Formatted IEs with Event include IE data fields. The data included in the format depends on the IE type. The data fields for each IE type is listed below. For more information on IE field values, refer to the *CSP Developer's Guide: Common Channel Signaling*.

0x01 Call Reference

The only length that is supported is 2:

Data[0] Most Significant Byte
 Data[1] Least Significant Byte

0x02 Called Party Number

Data[0] Type of Number
 Data[1] Numbering Plan Identification
 Data[2] Number of Digits (maximum is 28)
 Data[3–N] Number Digits (IA5/ASCII)

0x03 Calling Party Number

Data[0] Type of Number
Data[1] Numbering Plan Identification
Data[2] Presentation Indicator
Data[3] Screening Indicator
Data[4] Number of Digits (maximum is 28)
Data[5–N] Number Digits (IA5/ASCII)

0x04 Called Party Subaddress

Data[0] Type of Subaddress
Data[1] Number of Digits (maximum is 28)
Data[2–N] Number Digits (IA5/ASCII)

0x05 Calling Party Subaddress

Data[0] Type of Subaddress
Data[1] Number of Digits (maximum is 28)
Data[2–N] Number Digits (IA5/ASCII)

0x06 Redirecting Number

Data[0] Number Type
Data[1] Numbering Plan
Data[2] Presentation Indicator
Data[3] Screening Indicator
Data[4] Reason for Redirection
Data[5] Number of Digits
Data[6–N] Number Digits (IA5/ASCII)

0x07 Cause

Data[0] Coding Standard
Data[1] Location
Data[2] Recommendation
Data[3] Cause Value
Data[4] Diagnostics Length
Data[5–N] Diagnostics Data

0x08 Progress Indicator

Data[0] Coding Standard
Data[1] Location
Data[2] Progress Description

0x09 User-to-User

Data[0] Protocol Discriminator
Data[1–N] User Data

0x0A Codeset 6

Data[0] Raw IE Type
Data[1] Raw IE Length
Data[2–N] Raw IE Data
IE types are network- and protocol-specific.

0x0B Codeset 7

Data[0] Raw IE Type
Data[1] Raw IE Length
Data[2–N] Raw IE Data
IE types are network- and protocol-specific.

0x11 Codeset 5

The format for this IE is as follows:

Data[0] Raw IE Type
Data[1] Raw IE Length
Data[2–N] Raw IE Data
IE types are network- and protocol-specific.

0x12 Vari-A-Bill

Data[0] Invoke_ID (A unique identifier that specifies the feature request.)
Data[1] Billing Type (If the billing type is “Free call,” you do not need to include the billing data.)
Data[2] Hundreds
Data[3] Tens
Data[4] Ones
Data[5] Tenths
Data[6] Hundredths

Bearer Capability

You must represent this field as a Raw IE. Refer to the Q.931 specification for values.

Network-specific Facilities

You must represent this field as a Raw IE. Refer to the Q.931 specification for values.

0x13 Directory Number (DN) Provisioning

Data[0] Number of Digits in DN (n)
 Data[1] Digit 0
 Data[2] Digit 1
 :
 Data[n] Digit n
 Data[n+1] B Channel Access:
 Bit 0 Access to B1
 Bit 1 Access to B2
 Bit 2 Access to B1 and B2
 Bits 3-7 Reserved, must be 0x00

V5 Information Elements

V5 formatted IEs with Event include IE data fields. The data included in the format depends on the IE type. The data fields for each IE type are listed below. For more information on IE field values, refer to the *Developer's Guide: Common Channel Signaling*.

V5 PSTN IEs**0x01 Pulse Notification**

Data[0] Type 0x01
 Data[1] Length 0x00

0x02 Line Information

Data[0] Type 0x02
 Data[1] Length 0x01
 Data[2] Parameter

0x03 State

Data[0] Type 0x03
 Data[1] Length 0x01
 Data[2] PSTN FSM State

0x04 Autonomous Signaling Sequence

Data[0] Type 0x04
 Data[1] Length 0x01
 Data[2] Sequence Type

0x05 Sequence Response

Data[0] Type 0x05
Data[1] Length 0x01
Data[2] Sequence Response Value

0x06 Reserved**0x07 Cadence Ringing**

Data[0] Type 0x07
Data[1] Length 0x01
Data[2] Cadence Ringing Type

0x08 Pulsed Signal

Data[0] Type 0x08
Data[1] Length 0x01 or 0x05
Data[2] Pulse Type
Data[3] Pulse Duration Type (Optional)
Data[4] Suppression Indicator (Optional)
Data[5] Number of Pulses (Optional)
Data[6] Acknowledge Request Indicator (Optional)

0x09 Steady Signal

Data[0] Type 0x09
Data[1] Length 0x01
Data[2] Steady Signal Type

0x0A Digit Signal

Data[0] Type 0x0A
Data[1] Length 0x02
Data[2] Digit Acknowledge Request Indicator
Data[3] Digit Information

0x0B Recognition Time

Data[0] Type 0x0B
Data[1] Length 0x02
Data[2] Signal
Data[3] Duration Time

0x0C Pulsed Signal

Data[0]	Type	0x0C
Data[1]	Length	0x02 or 0x06
Data[2]	Signal	
Data[3]	Response	
Data[4]	Suppression Indicator (Optional, used only if response is a pulsed signal)	
Data[5]	Pulse Duration Type (Optional, used only if response is a pulsed signal)	
Data[6]	Acknowledge Request Indicator (Optional, used only if response is a pulsed signal)	
Data[7]	Number of Pulses (Optional, used only if response is a pulsed signal)	

0x0D Disable Autonomous Acknowledge

Data[0]	Type	0x0D
Data[1]	Length	0x01
Data[2]	Signal	

0x0E Cause

Data[0]	Type	0x0E
Data[1]	Length	N
Data[2]	Cause Type	
Data[3]	Diagnostic (Message Type ID)	
Data[4]	Diagnostic (Information Element ID)	

0x0F Cause

Data[0]	Type	0x0F
Data[1]	Length	N
...		Copy of IE with failed request

Parameter Type Values for Cadence Ringing Extended IE

0x02	CLIP CLI (Digits)
0x04	CLIP RAI (No Digits)
0x07	CNIP CNI (Characters)
0x08	CNIP RAN (No Characters)

The Cadence Ringing Extended IE can be encoded one of the following ways (A, B, C, or D)

**0x10 Cadence Ringing Extended
(Type A - Digits and Characters Present)**

Data[0] Type 0x10
 Data[1] Length
 Data[2] Cadence Ringing Type
 Data[3] Month (binary)
 Data[4] Day (binary)

 Data[5] Hour (binary)

 Data[6] Minute (binary)
 Data[7] CLIP CLI Parameter Type 0x02
 Data[8] CNIP CNI Parameter Type 0x07
 Data[9] Number of Digits in Calling Party

 Data[10] Calling Party Digit 1 (ASCII)

 Data[11] Calling Party Digit 2 (ASCII)

 : :
 Data[N - 1] Calling Party Digit N
 Data[N] Number of Characters in Calling Party
 Data[N + 1] Character 1
 Data[N + 2] Character 2
 Data[N + M] Character M

**0x10 Cadence Ringing Extended
(Type B - Digits Present, Characters Missing)**

Data[0] Type 0x10
 Data[1] Length
 Data[2] Cadence Ringing Type
 Data[3] Month (binary)
 Data[4] Day (binary)

 Data[5] Hour (binary)

 Data[6] Minute (binary)
 Data[7] CLIP CLI Parameter Type 0x02
 Data[8] CNIP RAN Parameter Type 0x08
 Data[9] Number of Digits in Calling Party

 Data[10] Calling Party Digit 1 (ASCII)

 Data[11] Calling Party Digit 2 (ASCII)

: :
 Data[N - 1] Calling Party Digit N
 Data[N] Reason for Absence of CNIP

**0x10 Cadence Ringing Extended
 (Type C - Digits Missing, Characters Present)**

Data[0] Type 0x10
 Data[1] Length
 Data[2] Cadence Ringing Type
 Data[3] Month (binary)
 Data[4] Day (binary)

 Data[5] Hour (binary)

 Data[6] Minute (binary)
 Data[7] CLIP RAI Parameter Type 0x04
 Data[8] CNIP CNI Parameter Type 0x07
 Data[9] Reason for Absence of Digits

 Data[10] Number of Characters in Calling Party

 Data[11] Calling Party Digit 2 (ASCII)
 Character 1
 Character 2
 Character M

**0x10 Cadence Ringing Extended
 (Type D - Digits and Characters Missing)**

Data[0] Type 0x10
 Data[1] Length
 Data[2] Cadence Ringing Type
 Data[3] Month (binary)
 Data[4] Day (binary)

 Data[5] Hour (binary)

 Data[6] Minute (binary)
 Data[7] CLIP RAI Parameter Type 0x04
 Data[8] CNIP RANParameter Type 0x08
 Reason for Absence of Digits*
 Reason for Absence of Characters*

*Reason for Absence Values:

0x4F Unavailable
 0x50 Private

0x11 Reserved**0x12 Called Party**

Data[0] Type 0x0E
 Data[1] Length N
 Data[2] Digit Acknowledge Request Indicator
 Data[3] Number of Digits
 Data[4] Digit[0] ASCII

 Data[5] Digit[1] ASCII

 : :
 Data[N+5] Digit[N] ASCII

V5 BCC IEs**0x15 User Port ID IE**

Data[0] Type 0x15
 Data[1] Length 0x02
 Data[2] User Port (MSB)
 Data[3] User Port (LSB)

0x17 V5 Timeslot ID IE

Data[0] Type 0x17
 Data[1] Length 0x03
 Data[2] Link ID
 Data[3] V5 Timeslot Number
 Data[4] Override (0 or 1)

0x1B V5 Timeslot ID IE

Data[0] Type 0x1B
 Data[1] Length 0x01
 Data[2] Reason
 0x00 Incomplete Normal
 0x01 AN Fault
 0x02 User port Not Provisional
 0x03 Invalid V5 Timeslot
 0x04 Invalid V5 Link ID
 0x05 Timeslot used as C Channel

IE Fields The list below shows the field values used in the ISDN Information Elements.

Billing Type

0x90 New Rate
0x91 Flat Rate
0x93 Premium Charge
0x94 Premium Credit
0x98 Free Call

Coding Standard

0x00 ITU-T
0x02 National Standard Coding
0x03 Standard Specific

Location

0x00 User
0x01 Private Network serving local user
0x02 Public Network serving local user
0x03 Transit Network
0x04 Public Network Serving Remote User
0x05 Private Network Serving Remote User
0x07 International Networks
0x0A Network Beyond Interworking Point

Number Plan Identification

0x00 Unknown
0x01 ISDN Numbering Plan/Recommendation E.164/E.163
0x02 Telephony numbering plan
0x09 Private numbering plan

Presentation Indicator

0x00 Presentation is allowed
0x01 Presentation is restricted
0x02 Number not available due to interworking

Progress Indicator

0x00 Unknown
0x01 Not End to End ISDN (In-band Info)
0x02 Destination Address Not ISDN
0x03 Origination Address Not ISDN
0x04 Call has Returned to the ISDN
0x08 In-band Information Available

Reason For Redirection

0x00 Unknown
0x01 Call Forward Busy
0x02 Call Forward No Answer
0x05 Called DTE Out of order
0x0A Call Forward by Called DTE
0x0F Unconditional Call Forwarding

Recommendation

0x00 Q.931
0x03 X.21
0x04 X.25

Screening Indicator

0x00 User-provided, not screened
0x01 User-provided, verified and passed
0x02 User-provided, verified and failed
0x03 Network-provided

Type of Number

0x00 Unknown
0x01 International number
0x02 National number
0x04 Subscriber number
0x06 Abbreviated number

Cause Codes This section lists all the Cause codes included in clearing messages (*Disconnect*, *Release*, *Release Complete*) and in error messages (*Status*).

For details on the cause code values, refer to Table I.2 in the Q.931 specification.

16	Normal Call Clearing. This is used in <i>Disconnect</i> if another cause is not specified in the clear request message.
17	User Busy
30	Response to <i>Status Query</i> message.
34	No channel available. This is used in <i>Release Complete</i> for rejecting <i>Setup</i> when no channel is available. The channel is indicated as preferred in the <i>Setup</i> message.
41	Temporary failure. This is used in <i>Disconnect</i> after channel purge because the host is not responding to the clear indication.
44	Requested channel not available. This is used in <i>Release Complete</i> to reject <i>Setup</i> when the specified channel is not available. Channel is indicated as exclusive in <i>Setup</i> .
65	Bearer capability not implemented. This is used in <i>Release Complete</i> to reject a <i>Setup</i> with invalid Bearer Capability.
96	Mandatory Information Element missing. Used in <i>Release Complete</i> or <i>Status</i> message
98	Invalid message type. This is used in the <i>Status</i> message.
100	Invalid Information Element content. This is used in the <i>Status</i> message.
102	Recovery on timer expiration. This is used in <i>Release</i> or <i>Release Complete</i> after expiration of timers T305 and T308.

DASS2/DPNSS Values

This section provides data values required for DASS2/DPNSS ICBs, including the following:

- Information Elements
- Cause Codes

Information Elements DASS2/DPNSS information elements may be sorted in the IE Library using the *ISDN Interface Configure* message. You may access them with Atomic Function 95 from the PPL design document.

0x01 Service Indicator Code (SIC)

The only length that is supported is 2:

Data[0] Octet 1 of the Service Indicator Code
 Data[1] Octet 2 of the Service Indicator Code (optional)

0x02 Selection Data

Data[0–N] per BTNR 190, Section 4 for DASS2
 and BTNR 188, Section 4 for DPNSS

0x03 Indication Data

Data[0–N] per BTNR 190, Section 4 for DASS2
 and BTNR 188, Section 4 for DPNSS

Cause Codes This sections lists all the Cause codes included in clearing messages (*Disconnect, Release, Release Complete*) and in error messages (*Status*).

For details on the cause code values, refer to Table I.2 in the Q.931 specification.

DASS2

The following are DASS2 cause codes:

0	Number Unobtainable
1	Address Incomplete
2	Network Termination
3	Service Unavailable
4	Sub Incompatible
5	Sub Changed Number
6	Invalid Request for Supplementary Service
7	Congestion
8	Subscriber Engaged
9	Sub Out of Service
10	Incoming Calls Barred
11	Outgoing Calls Barred
18	Remote Procedure Error
19	Service Incompatible
20	Acknowledgment
21	Signal Not Understood
26	Message Not Understood
30	NAE Error
31	No Reply from Sub
32	Service Termination
41	Access Barred
45	DTE Controlled Not Ready
46	DTE Uncontrolled Not Ready
48	Sub Call Termination
50	ET Isolated
51	Local Procedure Error

DPNSS

The following are DPNSS cause codes:

0	Number Unobtainable
1	Address Incomplete
2	Network Termination
3	Service Unavailable
4	Subscriber Incompatible
7	Congestion
8	Busy
9	Subscriber Out of Service
10	Incoming Calls Barred
19	Service Incompatible

20	Acknowledgment
21	Signal Not Understood
22	Signal Not Valid
23	Service Temporarily Unavailable
24	Facility Not Registered
25	Reject
26	Message Not Understood
27	Signaling System Incompatible
28	Route Out of Service
29	Transferred
30	NAE (Network Address Extension) Error
35	Channel Out of Service
36	Priority Forced Release
41	Access Barred
45	DTE Controlled Not Ready
46	DTE Uncontrolled Not Ready

1

4 Tag Length Value Blocks

Overview

Purpose This chapter lists the Tag Length Value Blocks (TLVs) used in the EXS API for the CSP. The single-byte (0xNN) TLVs are first, listed in hexadecimal numerical order. They are followed by the double-byte (0xNNNN) TLVs, also listed in hexadecimal numerical order.

Introduction to TLVs

In Call Control, the TLV is used to send data to, and receive data from, the CSP within API messages.

A TLV is an internal data structure consisting of the following elements:

- Tag - a label identifying the format of the TLV
- Length - the length of the data to follow
- Value - the data

Generic TLV Format The generic format of a TLV is shown below:

Byte	Description
1	Tag (MSB)
2	Tag (LSB)
3	Length (MSB)
4	Length (LSB)
5	Value[0] (Data)
:	:
:	Value[n]

Nested TLVs The Call Agent feature and the DSP/RTP feature use nested TLVs. The following provides an overview and example of nested TLVs. An API message can contain nested TLVs. These TLVs are nested into the value field of another TLV in order to capture multiple properties of a call within one TLV.

The following applies to these nested TLVs.

- The TLV Count fields in the API message accounts for the top level TLVs but not the nested TLVs.
- The same TLV can be used multiple times within the same message depending on the context of the nested TLVs.
- The Length field in any “parent” TLV counts the bytes in all of its nested TLVs.

Example:

```

00 69 00 43 00 EA 01
00 01 0D 03 00 12 06
    00 A7
    00 20
    01
    03
    00 33
    00 53
    00 01 \Number of TLVs - top level
    29 FF 00 4D \Parent TLV
Nested TLVs [ 2A 0E 00 04 D0 D1 2B 37
Nested TLVs [ 2A 01 00 41 \Length includes all nested TLVs
Nested TLVs [ 2A 03 00 01 00
Nested TLVs [ 2A 07 00 04 00 00 13 98
Nested TLVs [ 2A 0A 00 02 00 14
Nested TLVs [ 2A 02 00 13 \Length includes next 3 TLVs
Nested TLVs [ 2A 08 00 02 00 02
Nested TLVs [ 2A 09 00 01 02
Nested TLVs [ 2A 0B 00 04 00 00 1F 40
Nested TLVs [ 2A 02 00 13 \Length includes next 3 TLVs
Nested TLVs [ 2A 08 00 02 01 65
Nested TLVs [ 2A 09 00 01 32
Nested TLVs [ 2A 0B 00 04 00 00 1F 40
    
```

Interworking TLVs This chapter contains TLVs used for Interworking and are indicated as such. Some of these Interworking TLVs can also be used in Standard environments and that is indicated as well.

Refer to *Interworking in the Developer's Guide: Internet Protocol*.

Interworking TLVs are used in the following API messages:

Host to CSP

- *Outseize Control*
- *Route Control*
- *Inseize Control*
- *Connect with Data*
- *Release Channel with Data*
- *Outpulse Digits*
- *PPL Event Request*

CSP to Host

- *Request for Service with Data*
- *Channel Released with Data*
- *PPL Event Indication*
- *Channel Release Request*

TLVs

0x01 Trunk Number

Used in:

SS7 CLLI Configure message

Byte	Description
0	Tag 0x01
1	Length 0x04
2-5	Value[0-3] Trunk Number (IA5/ASCII characters)

0x01 Module, IP Address, and Subnet Mask

Use this TLV to configure the IP Address of a VoIP module.

In this TLV block, the module value (Data[0]) must change to reflect the IP address of the VoIP Module assigned. When IP addresses are re-assigned, the TCP/IP stack must be re-initialized. To ensure that the stack is re-initialized properly, you must reboot the board or module, using one of the following TLVs:

- Engage IP (0x02): If the Engage IP TLV is assigned, reset is automatic
- Reset (0x03)

Used in:

IP Address Configure message

IP Address Query message

Byte	Description
0	Tag 0x01 Module, IP Address, Subnet Mask
1	Length 0x09
2	Value[0] 0xFF (Main Board) 0x00 (VoIP Module 1) 0x01 (VoIP Module 2) 0x02 (VoIP Module 3) 0x03 (VoIP Module 4)
3-6	Value[1-4] IP Address (4 bytes)
7-10	Value[5-8] Subnet Mask (4 bytes)

0x01 Add Virtual Card

Used in:

Virtual Card Configure message

Byte	Description
0	Tag 0x01 Add Virtual Card
1	Length 0x02
2	Value[0] Slot Number 0x40- 0x5F
3	Value[1] Card Type 0xD2 Virtual T1 16 Span 0xD5 Virtual E1/J1 16 Span 0x80 Virtual 32 Span VDAC-ONE or IP Media Series 2

0x01 Client Socket Register by IP

Used in:

IP Socket Configure message

Byte	Description
0	Tag 0x01 Client Socket Register by IP
1	Length 0x0E
2	Value[0] 0x00 Module (0xFF = base board)
3	Value[1] Socket ID
4-7	Value[2-5] Local IP Address (4 bytes)
8, 9	Value[6, 7] Local Port
10-13	Value[8-11] Remote IP Address (4 bytes)
14, 15	Value[12, 13] Remote Port

0x02 Remove Virtual Card

Used in:

Virtual Card Configure message

Byte	Description
0	Tag 0x02 Remove Virtual Card
1	Length 0x01
2	Value[0] Slot Number 0x00- 0x0F

0x02 Server Socket Register by IP

Used in:

IP Socket Configure message

Byte	Description
0	Tag 0x02 Server Socket Register by IP
1	Length 0x0E
2	Value[0] (0x00) Module (0xFF = base board)
3	Value[1] Socket ID
4-7	Value[2-5] Local IP Address (4 bytes)
8, 9	Value[6, 7] Local Port
10-13	Value[8-11] Remote IP Address (4 bytes)
14, 15	Value[12, 13] Remote Port

0x02 Local CLLI

Used in:

SS7 CLLI Configure

Byte	Description
0	Tag 0x02
1	Length 0x0B
2-12	Value[0-10] Local CLLI (IA5/ASCII characters)

0x02 Engage IP

Used In:

IP Address Configure

Byte	Description
0	Tag 0x02 Engage IP
1	Length 0x00
2	Value Does Not Apply

0x03 Reset IP

Used In:

IP Address Configure

Byte	Description
0	Tag 0x03 Reset
1	Length 0x00
2	Value Does Not Apply

0x03 Client Socket Register by Name

Used in:

IP Socket Configure message

Byte	Description
0	Tag 0x03 Client Socket Register by Name
1	Length 0x00D
2	Value[0] 0x00 Module (0xFF = base board)
3	Value[1] Socket ID
4-7	Value[2-5] Local IP Address (4 bytes)
8	Value[6] Local Port, MSB
9	Value[7] Local Port, LSB
10	Value[8] Remote Name Length
11	Value[9] Remote Name[0]
12	Value[10] Remote Name[1]
13	Value[:] :
14	Value[:] Remote Port, MSB
15	Value[:] Remote Port, LSB

0x03 Remote CLLI

Used in:

SS7 CLLI Configure message*SS7 CLLI Query* message

Byte	Description
0	Tag 0x03
1	Length 0x0B
2-12	Value[0-10] Remote CLLI (IA5/ASCII characters)

0x04 Trunk Type

Used in:

SS7 CLLI Configure message*SS7 CLLI Query* message

Byte	Description
0	Tag 0x04
1	Length 0x01
2	Value[0] Trunk Type 0x00 Incoming One-way 0x01 Outgoing One-way 0x02 Two-way (default)

0x04 Server Socket Register by Name

Used in:

IP Socket Configure message

Byte	Description
0	Tag 0x04 Server Socket Register by Name
1	Length Variable
2	Value[0] 0x00 Module (0xFF = Main Board)
3	Value[1] Socket ID
4-7	Value[2-5] Local IP Address (4 bytes)
8, 9	Value[6, 7] Local Port
10	Value[8] Remote Name Length
11	Value[9] Remote Name[0]
12	Value[10] Remote Name[1]
13	Value[:]
:	Value[:] Remote Port
:	Value[:] Remote Port

0x04 Reset Indicator

At any time, the host can send a *Reset Configuration* message to the card to ensure that the IP address is assigned. The reset indicator value is returned in the Reset Indicator TLV. If the host sends a Reset or Engage TLV, an ACK is returned with all Indicator bits set to 0. If neither the Reset nor the Engage TLV is sent, the bit mask applies.

Used in:

IP Address Configure message

Byte	Description
0	Tag 0x04 Reset Indicator
1	Length 0x04
2	Value[0] 0x00 Reset Indicator, MSB
3	Value[1] Reset Indicator
4	Value[2] Reset Indicator
5	Value[3] Reset Indicator, LSB (0 = Disable, 1 = Enable) Bit 0: Main Board Bit 1: VoIP Module 1 Bit 2: VoIP Module 2 Bit 3: VoIP Module 3 Bit 4: VoIP Module 4 Bit 5: Reserved (0x00) Bit 6: Reserved (0x00) Bit 7: Reserved (0x00)

0x05 Gateway IP

The host can assign Gateway IP addresses the following ways:

- to the main board alone
- to the main board and all modules together
- to individual modules

If the first data byte of the Gateway IP TLV specifies the main board and all modules together (0x05), then the message must have a TLV block for the main board and for each module, specifying their individual IP addresses and subnet masks.

Important! Data byte 0x05 cannot be used in conjunction with 2 module VDAC-ONE or IP Network Interface Series 2 cards. For a two module VDAC-ONE or IP Network Interface Series 2 card, you must assign the gateway address with each individual module IP address.

Used in:

- *IP Address Configure* message
- *IP Address Query* message

Byte	Description
0	Tag 0x05 Gateway IP
1	Length 0x05
2	Value[0] 0xFF Main Board 0x00 VoIP Module 1 0x01 VoIP Module 2 0x02 VoIP Module 3 0x03 VoIP Module 4 0x05 All Modules, including Main Board
3-6	Value[1-4] IP Address (4 bytes)

0x05 Unregister Socket

Used in:

IP Socket Configure message

Byte	Description
0	Tag 0x05 Unregister Socket
1	Length 0x02
2	Value[0] 0x00 Module (0xFF = Main Board)
3	Value[1] Socket ID

0x06 Open Socket

Used in:

IP Socket Configure message

Byte	Description
0	Tag 0x06 Open Socket
1	Length 0x02
2	Value[0] 0x00 Module (0xFF = Main Board)
3	Value[1] Socket ID

0x07 Close Socket

Used in:

IP Socket Configure message

Byte	Description
0	Tag 0x07 Close Socket
1	Length 0x02
2	Value[0] 0x00 Module (0xFF = Main Board)
3	Value[1] Socket ID

0x08 Prohibit Traffic on Socket

Used in:

IP Socket Configure message

Byte	Description
0	Tag 0x08 Prohibit Traffic on Socket
1	Length 0x02
2	Value[0] 0x00 Module (0xFF = Main Board)
3	Value[1] Socket ID

0x09 Allow Traffic on Socket

Used in:

IP Socket Configure message

Byte	Description
0	Tag 0x09 Allow Traffic on Socket
1	Length 0x02
2	Value[0] 0x00 Module (0xFF = Main Board)
3	Value[1] Socket ID

0x09 Ethernet Link Redundancy

Use this TLV to enable Ethernet Link Redundancy. If you enable Ethernet Link Redundancy, the lower Ethernet Port is automatically activated. Note that Ethernet Link Redundancy is not supported on the SS7 Series 3 card.

Used in:

IP Address Configure message

IP Address Query message

Byte	Description
0	Tag 0x09 Ethernet Link Redundancy
1	Length 0x04
2-5	Value[0-3] 0x00000000 Disable 0x00000001 Enable

0x09 Ethernet Link Redundancy (DSP Series 2 Card)

Use this TLV to enable Ethernet Link Redundancy. If you enable Ethernet Link Redundancy, the upper Ethernet port is automatically activated.

Used in:

IP Address Configure message

IP Address Query message

Byte	Description
0	Tag 0x09 Ethernet Link Redundancy
1	Length 0x04
2-5	Value[0] 0x00000000 Disable 0x00000001 Enable

0x0A Activate Ethernet Port

Use this TLV to assign the active Ethernet Port on a VDAC-ONE card. The upper port is active by default. To activate the lower port, you must send this TLV. Note that Ethernet Link Redundancy is not supported on the SS7 Series 3 card.

Important! If you enable Ethernet Link Redundancy (TLV 0x09), the standby Ethernet Port is automatically activated upon failure of the active port.

This TLV also activates the Ethernet Interface of the Maxtrix Controller I/O as follows:

- Activate the Upper Port for Ethernet Interface B.
- Active the Lower Port for Ethernet Interface A.

Used in:

IP Address Configure message

IP Address Query message

Byte	Description
0	Tag 0x0A Activate Ethernet Port
1	Length 0x04
2	Value[0] 0x00000000 Upper Port Active 0x00000001 Lower Port Active

Important! The two TLVs above (0x09 and 0x0A) are returned in the response to a successful *IP Address Query* message.

0x0A Activate Ethernet Port (DSP Series 2 Card)

Use this TLV to assign the active Ethernet port for a DSP Series 2 card. If you enable Ethernet Link Redundancy (TLV 0x09), an available standby Ethernet Port is automatically activated upon failure of the active port. This TLV activates the Ethernet Interfaces on the Multi-Function Media I/O card as follows:

- Upper port (NET1)
- Middle port (NET2)
- Lower port (NET3)

Used in:

IP Address Configure message

IP Address Query message

Byte	Description
0	Tag 0x0A Activate Ethernet Port
1	Length 0x04
2-5	Value[0] 0x00000001 Upper Port Active (NET1) 0x00000002 Middle Port Active (NET2) 0x00000003 Lower Port Active (NET3)

The two TLVs above (0x09 and 0x0A) are returned in the response to a successful *IP Address Query* message. If an 0x00FF is returned as the value of the Active Ethernet Port TLV (0x0A) in the response

to a *IP Address Query* message, it indicates that Ethernet Link Redundancy is disabled.

0x0A Socket Status

Used in:

IP Socket Status Report message

Byte	Description
0	Tag 0x0A Socket Status
1	Length 0x03
2	Value[0] 0x00 Module (0xFF = Main Board)
3	Value[1] 0x01-0x02 Socket ID
4	Value[2] Socket Status 0x00 Socket is not connected, but allow for traffic when connected 0x01 Socket is connected. Allow for Traffic 0x02 Socket is not connected. Prohibit Traffic.

0x0B Socket ID

Used in:

0x00F1 *IP Socket Query* message

Byte	Description
0	Tag 0x0B Socket ID
1	Length 0x02
2	Value[0] 0x00 Module (0xFF = Main Board)
3	Value[1] Socket ID

0x0C Socket Information

Used in:

0x00F1 *IP Socket Query* message

Byte	Description
0	Tag 0x0C Socket Information
1	Length 0x014
2	Value[0] 0x00 Module (0xFF = Main Board)
3	Value[1] Socket ID
4	Value[2] Socket Status
5	Value[3] Socket Client Flag 0x00 Server Side Socket 0x01 Client Side Socket
6-9	Value[4-7] Local IP Address (4 bytes)
10, 11	Value[8, 9] Local Port
12-15	Value[10-13] Remote IP Address (4 bytes)
16, 17	Value[14, 15] Remote Port
18-21	Value[16-19] Socket User Name SASI Socket user is SASSI LHST Socket user is Local host communication module 0X00 00 00 00 Socket is not binded to any user

0x11 Signaling Route Test Control Status

Used in:

0x1E and 0x001E Generic PPL ICBs in *PPL Event Request* message

Byte	Description
0	Tag SRTC Status
1	Length 0x01
2	Value[0] 0x11

0x12 Signaling Route Test Control DPC

Used in:

0x1E and 0x001E Generic PPL ICBs in *PPL Event Request* message

Byte	Description
0	Tag 0x12 Signaling Route Test Control DPC
1	Length 0x01
2	Value[0] 0x02

0x14 Reset VDAC Module

Used in:

IP Address Configure message

Byte	Description
0	Tag 0x14
1	Length 0x01
2	Value[0] Module VDAC-ONE 0-3 IPN-2 0-1

0x6F L3 IP Protocol Type

Used in:

PPL Configure message

Byte	Description
0, 1	Tag 0x006F
2, 3	Length 0x0001
4	Value[0] 0x08 SIP 0x09 H.323

0x0001 Add IP Signaling Series 3 Card

Used in:

Matrix Configure message (0x7D)*IP Signaling Series 3 Card Configure* message (0x0100)

Byte	Description
0, 1	Tag 0x0001
2, 3	Length 0x0002
4	Value[0] IP Signaling Series 3 Card ID, MSB0x00
5	Value[1] IP Signaling Series 3 Card ID, LSB0x00-0x3F

0x0001 Create V5 ID

Used in:

V5 Configure message*V5 Configuration Query* message

Byte	Description
0	Tag 0x00 Create V5 ID (MSB)
1	Tag 0x01 Create V5 ID (LSB)
2	Length 0x00 (MSB)
3	Length 0x02 (LSB)
4	V5 ID 0x00 (MSB)
5	V5 ID 0x00 - 0x0F (LSB)

0x0002 Destroy V5 ID

Used in:

V5 Configure message

Byte	Description
0	Tag 0x00 Destroy V5 ID (MSB)
1	Tag 0x02 Destroy V5 ID (LSB)
2	Length 0x00 (MSB)
3	Length 0x00 (LSB)

0x0002 Remove IP Signaling Series 3 Card

Used in:

Matrix Configure message (0x7D)*IP Signaling Series 3 Card Configure* message (0x0100)

Byte	Description
0, 1	Tag 0x0002
2, 3	Length 0x0002
4	Value[0] IP Signaling Series 3 card ID to be removed, MSB: 0x00 Remove a specific IP Signaling Series 3 card 0xFF Remove every configured IP Signaling Series 3 cards
3	Value[1] IP Signaling Series 3 card ID to be removed, LSB: 0x00-0x3F IP Signaling Series 3 card to be removed 0xFF Remove all configured IP Signaling Series 3 cards

0x0003 Matrix ID

Used in:

IP Signaling Series 3 Card Configure message*IP Signaling Series 3 Card Query* message

Byte	Description
0, 1	Tag 0x0003
2, 3	Length 0x0002
4	Value[1] Matrix ID, MSB 0x00
5	Value[2] Matrix ID, LSB 0x00 - 0x3F

0x0003 Protocol Type

Used in:

Matrix Configure message (0x007D)*Matrix Query* message (0x0097)*Server Configuration* message (0x0100)*Matrix Status Report* message (0xE5)

Byte	Description
0, 1	Tag 0x0003
2, 3	Length 0x0004
4	Value[0] Protocol, MSB 0x00
5	Value[1] Protocol, LSB 0x01 H.323
6	Value[2] Platform, MSB 0x00
7	Value[3] Platform, LSB 0x00 CSP 0x01 Solaris 0x02 Windows NT® 0x03 HPUX

0x0003 V5 Variant ID

Used in:

V5 Configure message*V5 Configuration Query* message

Byte	Description
0	Tag 0x00 V5 Variant ID (MSB)
1	Tag 0x03 V5 Variant ID (LSB)
2	Length 0x00 (MSB)
3	Length 0x01 (LSB)
4	Value[0-127] 0x00 V5 Variant ID Value

0x0004 V5 Variant Type

Used in:

V5 Configure message*V5 Configuration Query* message

Byte	Description
0	Tag 0x00 V5 Variant Type (MSB)
1	Tag 0x04 V5 Variant Type (LSB)
2	Length 0x00 (MSB)
3	Length 0x01 (LSB)
4	Variant Type 0x01 0x02 = V5.2 (default)

0x0004 Remove Matrix

Used in:

IP Signaling Series 3 Card Configure message (0x0100)

Byte	Description
0, 1	Tag 0x0004
2, 3	Length 0x0002
4	Value[0] Call Server ID to be removed, MSB 0x00 To remove a specific Matrix 0xFF To remove all configured Matrices
5	Value[1] Call Server ID to be removed, LSB 0x00-0x3F To remove a specific Matrix 0xFF To remove all configured Call Matrices

0x0004 Configuration Tag

Used in:

Matrix Configure message (0x007D)*Matrix Query* message (0x0097)

Byte	Description
0, 1	Tag 0x0004
2, 3	Length 0x0002
4, 5	Value[0-1] Configuration Tag 0 - 0xFFFF

0x0005 IP Signaling Series 3 Card IP Address

Used in:

Matrix Configure message (0x007D)

Matrix Query message (0x0097)

Byte	Description
0, 1	Tag 0x0005
2, 3	Length 0x0006
4-7	Value[0-3] IP Address (4 bytes)
8, 9	Value[4, 5] Port Number (Variable)

0x0005 IP Signaling Series 3 Card Slot Number

Used in:

IP Signaling Series 3 Card Configure message (0x0100)

IP Signaling Series 3 Card Query message (0x0101)

Byte	Description
0, 1	Tag 0x0005
2, 3	Length 0x0002
4	Value[0] IP Signaling Series 3 Card Slot Number, MSB 0x00
5	Value[1] IP Signaling Series 3 Card Slot Number, LSB 0x00-0x0F

0x0005 V5 Side

Used in:

V5 Configure message

V5 Configuration Query message

Byte	Description
0	Tag 0x00 V5 LE or AN Side (MSB)
1	Tag 0x05 V5 LE or AN Side (LSB)
2	Length 0x00 (MSB)
3	Length 0x01 (LSB)
4	LE or AN 0x00 = LE (default) 0x01 = AN (AN not supported)

0x0006 Route Group ID

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

0x1E Generic PPL ICB in *Route Control* message

Byte	Description
0, 1	Tag 0x0006
2, 3	Length 0x0002
4, 5	Value[0-1] Route Group ID or 0x0000 Null

0x0006 Add V5 User Port Range

Used in:

V5 Configure message

V5 Configuration Query message

Byte	Description
0	Tag 0x00 Add V5.2 User Port Range (MSB)
1	Tag 0x06 Add V5.2 User Port Range (LSB)
2	Length 0x00 (MSB)
3	Length 0xNN (LSB)
4	Number of Ranges
5	User Port Type
6	Start User Port ID, MSB
7	Start User Port ID, LSB
8	End User Port ID, MSB
9	End User Port ID, LSB
:	:
:	Start User Port ID, MSB
:	Start User Port ID, LSB
:	End User Port ID, MSB
:	End User Port ID, LSB
:	:

0x0007 Remove V5 User Port Range

Used in:
V5 Configure message

Byte	Description
0	Tag 0x00 Remove V5.2 User Port Range (MSB)
1	Tag 0x07 Remove V5.2 User Port Range (MSB)
2	Length 0x00 (MSB)
3	Length 0xNN (LSB)
4	Number of Ranges
5	User Port Type
6	Start User Port ID, MSB
7	Start User Port ID, LSB
8	End User Port ID, MSB
9	End User Port ID, LSB
:	:
:	Start User Port ID, MSB
:	Start User Port ID, LSB
:	End User Port ID, MSB
:	End User Port ID, LSB
:	:

0x0007 IP Signaling Series 3 Card Service State

Used in:
Matrix Configure message (0x007D)

Byte	Description
0, 1	Tag 0x0007
2, 3	Length 0x0002
4	Service, MSB 0x00
5	Service, LSB 0x00 Out of Service 0x01 In Service

0x0007 Incoming Span/Channel, Single Address

Used in:
PPL Table Download (0x00D6)

Byte	Description
0, 1	Tag 0x0007
2, 3	Length 0x0006
4, 5	Value[0,1] Span

Byte	Description
6	Value[2] Channel
7-9	Value[3-5] Mask

0x0007 Incoming Span/Channel, Range Address

Used in:

PPL Table Download (0x00D6)

Byte	Description
0, 1	Tag 0x0007
2, 3	Length 0x0009
4, 5	Value[0,1] Span
6	Value[2] Channel
7, 8	Value[3,4] To Span
9	Value[5] To Channel
10-12	Value[6-8] Mask

0x0008 IP Signaling Series 3 Card Compatibility Tag

Used in:

Matrix Configure message (0x7D)

IP Signaling Series 3 Card Query message (0x0101)

IP Signaling Series 3 Card Configure message (0x0100)

Byte	Description
0, 1	Tag 0x0008
2, 3	Length 0x0002
4	Value[0] Value set up on both sides of the interface, MSB 0x00-0xFF
5	Value[1] Value set on both sides of the interface, LSB 0x00 - 0xFF

0x0008 Criteria Type

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

0x1E and 0x001E Generic PPL ICBs in *Route Control* message

Byte	Description
0, 1	Tag 0x0008 Criteria Type
2, 3	Length 0x0002
4, 5	Value[0, 1] <ul style="list-style-type: none"> 0x0000 Null 0x0001 - 0004 Address Digits 1, 2, 3, 4 0x0005 Time of day 0x0007 Incoming Channel 0x0010 Incoming Resource Group 0x001A Incoming Node 0x0065 Physical VoIP Channel 0x0071 Virtual VoIP Channel 0x03E9 - 03F8 User Defined 0x2710 - 0x4E20 Universal Protocol Data 0x2792 Source IP Address 0x27B4 IP Address Criteria 0x2A0E Media Connection Address

0x0009 Address Element TLV

Important! In the Address Element TLV, the *Resource Attribute Configure* message requires either the Expanded Span Channel AIB or the IP Address AIB, but not both. The Address Element Blocks TLV that follows can refer to either the Expanded Span/Channel AIB or to the IP Address AIB.

For Expanded Span/Channel AIB:

Use this TLV to address configuration to a specific channel. If you are changing the attributes of an existing connection, use the Channel address type (0x0D).

Used in:
Resource Attribute Configure message

Byte	Description
0, 1	Tag: Address Element (0x0009)
2, 3	Length: 0x0005
4	Value[0] AIB Type: 0x0D (Expanded Span/Channel)
5	Value[1] AIB Length (0x03)
6	Value[2] Logical Span (MSB)
7	Value[3] Logical Span (LSB)
8	Value[4] Channel

For IP Address AIB:

Use this TLV to address configuration to a specific VoIP module.

Important! This TLV is required for all attribute configuration except for Gateway Mode (Gateway Mode TLV, 0x01D0), which configures an entire IP Network Interface Series 2 card.

If you are pre-configuring the attributes on a VoIP module, use the IP Address type (0x3E).

Used in:

Resource Attribute Configure message

Byte	Description
0, 1	Tag: Address Element (0x0009)
2, 3	Length: 0x0006
4	Value: Data[0] AIB Type: 0x3E (IP Address)
5	Value: Data[1] AIB Length 0x04
6-9	Value: Data[2-5] IP Address (4 bytes)

0x0009 Poll Interval

The poll interval, which is in 10 milliseconds units, should be configured with values greater than 5. If configured with anything less than 5, the IP Signaling Series 3 card will automatically set the poll interval to 5.

Important! This poll is the Matrix to IP Signaling Series 3 card poll not the IP Signaling Series 3 card to Host poll.

Used in:
Matrix Configure message (0x7D)

Byte	Description
0, 1	Tag 0x0009
2, 3	Length 0x0004
4, 5	Value[0-1] Number of Seconds Between Polls 0x00 - 0xFF In 10 milliseconds units
6, 7	Value[2-3] Number of Missed Polls Before Link is Declared Down 0x00 - 0xFF

0x0009 Terminating Channel

Use this TLV to specify a destination channel or a node. This TLV is used for the Terminating Channel routing method. The span/channel or node are specified in the TLV in an Address Information Block (AIB).

Used in:
 0x001E or 0x1E Generic PPL ICB in *Route Control* message

Byte	Description
0, 1	Tag 0x0009 Terminating Channel or Node
2, 3	Length 0x0005
4	Value[0] AIB Type: 0x0D
5	Value[1] AIB Length (0x03)
6	Value[2] Logical Span (MSB)
7	Value[3] Logical Span (LSB)
8	Value[4] Channel

0x0009 Terminating Span/Channel, Single Address

Used in:
PPL Table Download (0x00D6)

Byte	Description
0, 1	Tag 0x0009
2, 3	Length 0x0006
4, 5	Value[0,1] Span
6	Value[2] Channel
7-9	Value[3-5] Mask

0x0009 Terminating Span/Channel, Range Address

Used in:

PPL Table Download (0x00D6)

Byte	Description
0, 1	Tag 0x0009
2, 3	Length 0x0009
4, 5	Value[0,1] Span
6	Value[2] Channel
7, 8	Value[3,4] To Span
9	Value[5] To Channel
10-12	Value[6-8] Mask

0x000A Matrix IP Address

Used in:

*Matrix Configure message (0x007D)**Matrix Query message (0x0097)*

Byte	Description
0, 1	Tag 0x000A
2, 3	Length 0x0006
4-7	Value[0-3] IP Address (4 bytes)
8, 9	Value[4-5] Port Number

0x000A Add Poll Interval

The poll interval, which is in 10 milliseconds, should be configured with values greater than 5. If configured with anything less than 5, the IP Signaling Series 3 card will automatically set the poll interval to 5.

Important! This poll is the IP Signaling Series 3 card to Matrix poll not the IP Signaling Series 3 card to Host poll.

Used in:

IP Signaling Series 3 Card Configure message (0x0100)

IP Signaling Series 3 Card Query message (0x0101)

Byte	Description
0, 1	Tag 0x000A
2, 3	Length 0x0004
4, 5	Value[0-1] Number of Seconds Between Polls 0x00 - 0xFF In 10 millisecond units
6, 7	Value[2-3] Number of Missed Polls Before Link is Declared Down 0x00 - 0xFF

0x000A V5 C Channel Assignment

Used in:

V5 Configure message

V5 Configuration Query message

Byte	Description
0	Tag 0x00 V5 C Channel Assignment (MSB)
1	Tag 0x0A V5 C Channel Assignment (LSB)
2	Length 0x00 (MSB)
3	Length 0x06 (LSB)
4	Type 0x01 = PG1 Primary 0x02 = PG1 Secondary 0x03 = PG#2 (Not Supported)
5	Logical Span ID for Primary Link, MSB
6	Logical Span ID for Primary Link, LSB
7	Primary Physical Channel 16
8	V5 C Channel ID 0x00 = Primary 0x01 - 0x2B = Others
	Link ID

0x000B V5 Cpath Assignment

Used in:

V5 Configure message

V5 Configuration Query message

This TLV is currently not supported.

Byte	Description
0	Tag 0x00 V5 Cpath Assignment (MSB)
1	Tag 0x0B V5 Cpath Assignment (LSB)
2	Length 0x00 (MSB)
3	Length 0x02 (LSB)
4	C Channel ID
5	Cpath Type 0x00 = PSTN

Before adding a C path you must create a C Channel and the C Channel should belong to Protection Group 2. Protection Group 2 is not supported.

0x000B IP Signaling Series 3 Card Version

Used in:

Matrix Configure message (0x007D)

Matrix Status Report message

Matrix Query message (0x0097)

Byte	Description
0, 1	Tag 0x000B
2, 3	Length 0x0006
4, 5	Major Revision Variable
6, 7	Minor Revision Variable
8, 9	Build Number Variable

0x000C Matrix IP Address

Used in:

IP Signaling Series 3 Card Configure message (0x0100)

IP Signaling Series 3 Card Query message (0x0101)

Byte	Description
0, 1	Tag 0x000C
2, 3	Length 0x0006
4-7	Value[0-3] IP Address (4 Bytes)
8, 9	Value[4, 5] Port Number

0x000C V5 Add Link

Used in:

V5 Configure message*V5 Configuration Query* message

Byte	Description
0	Tag 0x00 V5 Add Link (MSB)
1	Tag 0x0C V5 Add Link (LSB)
2	Length 0x00 (MSB)
3	Length 0x04 (LSB)
4	Logical Span ID for Link, MSB 0x00
5	Logical Span ID for Link, LSB 0x00
6	V5 Link ID (0 - 15) 0x00
7	Real V5 Link ID (0 - 255) 0x00

0x000D V5 Remove Link

Used in:

V5 Configure message

Byte	Description
0	Tag 0x00 V5 Remove Link (MSB)
1	Tag 0x0D V5 Remove Link (LSB)
2	Length 0x00 (MSB)
3	Length 0x01 (LSB)
4	V5 Link ID 0x00 - 0x0F

0x000E Terminating Span/Channel Offset

Used in:

0x1E Generic PPL ICB in *Route Control* message

Byte	Description
0, 1	Tag 0x000E Terminating Span/Channel Offset
2, 3	Length 0x0003
4	Value[0] Offset

0x000E IP Signaling Series 3 Card Interface Service State

Used in:

IP Signaling Series 3 Card Configure message (0x0100)

IP Signaling Series 3 Card Query message (0x0101)

Byte	Description
0, 1	Tag 0x000E
2, 3	Length 0x0002
4	Value[0] State, MSB 0x00
5	Value[1] State, LSB 0x00 Out of Service 0x01 In Service

0x000F IP Signaling Series 3 Card Interface Support

Used in:

IP Signaling Series 3 Card Configure message (0x0100)

Byte	Description
0, 1	Tag 0x000F
2, 3	Length 0x0010
4	Value[0] L3/L4 Signaling Interface, MSB 0x00
5	Value[1] L3/L4 Signaling Interface, LSB 0x00 Disable 0x01 Enable
6	Value[2] LX/ENG Connection Interface, MSB 0x00
7	Value[3] LX/ENG Connection Interface, LSB 0x00 Disable 0x01 Enable
8-20	Value[4-16] Reserved 0x00 (Disabled)

0x000F Router Protocol ID

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

0x1E and 0x001E Generic PPL ICBs in *Route Control* message

Byte	Description
0, 1	Tag 0x000F Router Protocol ID
2, 3	Length 0x0001
4	Value[0] Router Protocol ID 0x0B (default)

0x000F V5 Remove C Channel Assignment

Used in:

V5 Configure message

Byte	Description
0	Tag 0x00 V5 Remove C Channel Assignment (MSB)
1	Tag 0x0F V5 Remove C Channel Assignment (LSB)
2	Length 0x00 (MSB)
3	Length 0x01 (LSB)
4	V5 C Channel ID 0x00 = Primary 0x01 - 0x2B = Others

0x000F Link Down Timer

This timer is used when the link between the IP Signaling Series 3 Card and the CSP Matrix Series 3 Card goes down. If there is a secondary CSP Matrix Series 3 Card present in the node, the IP Signaling Series 3 card triggers this timer and does not release any active calls (calls in the answered state). The IP Signaling Series 3 card waits until the timer expires to re-establish the connection with the secondary matrix.

If the connection is re-established, all the active calls stay connected; however, no new calls (incoming or outgoing) can be made when the link between the CSP Matrix Series 3 Card and IP Signaling Series 3 card is temporarily down.

If the link is not re-established or if there is no secondary CSP Matrix Series 3 Card in the node, all of the active calls are released on the H.323 (IP) end.

Used in:

IP Signaling Series 3 Card Configure message*IP Signaling Series 3 Card Query* message

Byte	Description
0, 1	Tag Link Down Timer
2, 3	Length 0x02
4, 5	Value[0, 1] Timer in 10 ms intervals (MSB, LSB). 0x2EE0 (Default is 12000 seconds - 2 minutes)

0x0010 Incoming Resource Group

Used in:

0x1E Generic PPL ICB in *Route Control* message

Byte	Description
0, 1	Tag 0x0010 Incoming Resource Group
2, 3	Length 0x0002
4, 5	Value[0, 1] Incoming Resource Group ID [MSB, LSB]

0x0010 V5 Interface ID

The V5 Interface ID is the actual ID used in the network.

Used in:

V5 Configure message

V5 Configuration Query message

Byte	Description
0, 1	Tag 0x0010 V5 Interface ID (actual ID used in the network)
2, 3	Length 0x0003
4	V5 Interface ID, MSB 0x00
5	V5 Interface 0x00
6	V5 Interface ID, LSB 0x00

0x0010 Incoming Res Group ID, Single Address

Used in:

PPL Table Download (0x00D6)

Byte	Description
0, 1	Tag 0x0010
2, 3	Length 0x0004
4, 5	Value[0,1] Node
6, 7	Value[2,3] Mask

0x0010 Incoming Res Group ID, Range Address

Used in:

PPL Table Download (0x00D6)

Byte	Description
0, 1	Tag 0x0010
2, 3	Length 0x0006

Byte	Description
4, 5	Value[0,1] From Node
6, 7	Value[2,3] To Node
8, 9	Value[4,5] Mask

0x0012 Remove V5 Cpath Assignment

Used in:

V5 Configure message

This TLV is not currently supported.

Byte	Description
0, 1	Tag 0x0012 Remove V5 C path Assignment
2, 3	Length 0x0001
4	C Channel ID 0x01 = V5 ID 0x03 = C Channel ID

0x0013 Routing Method

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

0x1E and 0x001E Generic PPL ICBs *Route Control* message

Byte	Description
0, 1	Tag 0x0013 Routing Method
2, 3	Length 0x0002
4, 5	Value[0-1] 0x0000 Null 0x0006 Route Group ID 0x0008 Criteria Type 0x0009 Terminating Channel 0x0010 Incoming Resource Group 0x0015 Terminating Resource Group 0x000E Terminating Span/Channel Offset 0x001A Incoming Node

0x0014 Message Header Information

Used in:

PPL Event Indication message

Byte	Description
0, 1	Tag 0x0014
2, 3	Length 0x0004
4, 5	Value[0-1] Contains the message type 0x00D8 - Route Control or 0x002C - Outseize Control
5,6	Value[2-3] Contains the two-byte sequence number of the Route Control /Outseize Control message sent by the host. The valid values ranges from 0x0000-0xFFFF.

0x0015 Terminating Resource Group

Used in:

0x1E Generic PPL ICB in *Route Control* message

Byte	Description
0, 1	Tag 0x0015 Terminating Resource Group
2, 3	Length 0x0002
4, 5	Value[0-1] Resource Group ID [MSB, LSB]

0x0015 Terminating Resource Group ID, Single Address

Used in:

PPL Table Download (0x00D6)

Byte	Description
0, 1	Tag 0x0015
2, 3	Length 0x0004
4, 5	Value[0,1] Node
6, 7	Value[2,3] Mask

0x0015 Terminating Resource Group ID, Range Address

Used in:

PPL Table Download (0x00D6)

Byte	Description
0, 1	Tag 0x0015
2, 3	Length 0x0006
4, 5	Value[0,1] From Node
6, 7	Value[2,3] To Node
8, 9	Value[4,5] Mask

0x001A Incoming Node

Used in:

0x1E Generic PPL ICB in *Route Control* message

Byte	Description
0, 1	Tag 0x001A Incoming Node
2, 3	Length 0x01
4, 5	Value[0, 1] Node ID

0x001D Special Routing Instruction

Byte	Description
0, 1	Tag 0x001D Special Routing Instruction
2, 3	Length 0x02
4, 5	Value[0, 1] 0x0003 Channel Reservation

0x0020 IP Signaling Series 3 Card Interface Status

Used in:

IP Signaling Series 3 Card Status Report message (0x0105)

Byte	Description
0, 1	Tag 0x0020
2, 3	Length 0x0004
4	Value[0] DSM FSM State Values, MSB 0x00
5	Value[1] DSM FSM State Values, LSB 0xFF FSM uninitialized 0x01 FSM initialized 0x02 FSM waiting for socket accessible indication 0x03 FSM waiting for socket inaccessible indication 0x04 FSM waiting for status request from the call server 0x05 FSM waiting for channel status request from the call server 0x06 FSM active 0x07 FSM inactive
6	Value[2] Call Server Values, MSB 0x00
7	Value[3] Call Server Values, LSB 0x01 Call server socket inaccessible 0x02 Call server socket accessible 0x03 Call server open failed 0x04 Call server close failed 0x05 Call server link down

0x0021 IP Signaling Series 3 Card Host Poll

Used in:

IP Signaling Series 3 Card Host Poll message (0x0104)

Byte	Description
0, 1	Tag 0x0021
2, 3	Length 0x0009
4, 5	Value[0, 1] IP Signaling Series 3 Card Status 0x0010
6	Value[2] System Type 0x10 IP Signaling Series 3 Card
7	Value[3] IP Signaling Series 3 Card Type 0x01 H.323 IP Signaling Series 3 Card
8	Value[4] IP Signaling Series 3 Card State 0x01 Booting 0x02 Initializing 0x03 Standby 0x04 Not used 0x05 Active
9	Value[5] Reserved
10	Value[6] IP Signaling Series 3 Card Status Bits Bit 0 Host Messaging 0x00 OK 0x01 Not receiving messages Bits 1 through 7 - not used
11, 12	Value[7, 8] Reserved

0x0051 Dialog Terminated

Used in:

SS7 TCAP Parameters ICB in PPL Event Indication

Byte	Description
0, 1	Tag 0x0051
2, 3	Length 0x0001
4	Value[0] Dialog Terminated

0x005E Resource Client ID

Used in:
Route Control message

Byte	Description
0, 1	Tag 0x005E
2, 3	Length 0x0008
4,5	Value[0, 1] Alias ID (MSB, LSB) 0x0002 = V5 Client
6,7	Value[2,3] Reserved
8-11	Value[4-7] Client ID: V5 ID

0x0061 Reject Reason

Used in:
PPL Event Indication message

Byte	Description
0, 1	Tag 0x0060
2, 3	Length 0x0002
4, 5	Value[0, 1] Reject Reason 0x02 Admission Reject (ARJ) Indication 00 01 Called Party Not Registered 00 02 Invalid Permission 00 03 Request Denied 00 04 Undefined Reason 00 05 Caller Not Registered 00 06 Route Call To Gatekeeper 00 07 Invalid Endpoint Identifier 00 08 Resource Unavailable 00 09 Security Denial 00 0A QOS Control Not Supported 00 0B Incomplete Address 0x05 Bandwidth Reject (BRJ) Indication 00 01 Not Bound 00 02 Invalid Conference ID 00 03 Invalid Permission 00 04 Insufficient Resources 00 05 Invalid Revision 00 06 Undefined Reason

0x0062 Resource Group Reorder Method

Byte	Description
0, 1	Tag 0x0062
2, 3	Length 0x0001
4	Value[0] 0x01 - Round Robin 0x02 - Least Recently Used (LRU)

0x0063 Route Number

Byte	Description
0, 1	Tag 0x0063
2, 3	Length 0x0003
4	Value[0] 0x01 - Route Table ID
5,6	Value[1,2] Route Number

0x0065 Physical VoIP Channel

Indicates to the router to hunt a physical VoIP channel.

Used in:

PPL Table Download message

0x1E Generic PPL ICB in *Route Control* message

Byte	Description
0, 1	Tag 0x0065
2, 3	Length 0x0002
4,5	Value[0-1] 0x0000

0x006D SCCP Return Reason

Used in:

PPL Event Indication message

Byte	Description
0, 1	Tag 0x006D
2, 3	Length 0x0001
4	Value[0] Reason Number 0x00 Translation does not exist 0x03 Subsystem failure 0x04 Unequipped user 0x05 MTP failure 0x06 Network congestion

0x0071 Virtual VoIP Channel

Indicates to the router that a virtual VoIP channel will be hunted.

Used in:

PPL Table Download message

0x1E and 0x001E Generic PPL ICBs in *Route Control* message

Byte	Description
0, 1	Tag 0x0071
2, 3	Length 0x0002
4,5	Value[0-1] 0x0000

0x0076 SCCP Other Return Reason

Used in:

PPL Event Indication API message

Byte	Description
0, 1	Tag 0x0076
2, 3	Length 0x0001
4	Value[0] Reason Number 0x00 SCCP message syntax error 0x01 Unsupported message (connection-oriented)

0x0080 IP Signaling Series 3 Card Status

Used in:

Matrix Status Report message (0x00E5)

Byte	Description
0, 1	Tag 0x0080
2, 3	Length 0x0004
4, 5	Value[0, 1] DSM FM State Values 0x00FF FSM uninitialized 0x0001 FSM initialized 0x0002 FSM waiting for Socket Accessible Indication 0x0003 FSM waiting for Socket Inaccessible Indication 0x0004 FSM waiting for Status Indication from the IP Signaling Series 3 Card 0x0005 FSM waiting for Span Status query 0x0006 FSM waiting for Span Status change indication 0x0007 FSM waiting for Channel Status Indication from the IP Signaling Series 3 Card 0x0008 FSM waiting for Channel Status Complete from the IP Signaling Series 3 Card 0x0009 FSM active 0x000A FSM inactive
6, 7	Value[2, 3] IP Signaling Series 3 Card Status Information 0x0000 IP Signaling Series 3 Card status information (received device server status indication) 0x0001 Type did not authenticate 0x0002 Configuration Tag did not authenticate 0x0003 Compatibility Tag did not authenticate 0x0004 IP Signaling Series 3 card socket inaccessible 0x0005 IP Signaling Series 3 card socket accessible (at least one active socket to IP Signaling Series 3 card) 0x0006 Socket open failure 0x0007 Socket close failure 0x0008 IP Signaling Series 3 card active IP Signaling Series 3 available to process calls) 0x0009 IP Signaling Series 3 card link down (socket connection is open, but the IP Signaling Series 3 card is not responding to poll requests)

0x0100 RTP Payload Type (VDAC-ONE)

Used in:

0x1E and 0x001E Generic PPL ICBs in:

Outseize Control message

Route Control message

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description																																						
0, 1	Tag 0x0100 RTP Payload Type																																						
2, 3	Length 0x0001																																						
4	Value[0] <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;"><u>Payload Type</u></th> <th style="text-align: left;"><u>Basic Packet Rate</u></th> </tr> </thead> <tbody> <tr><td>0x00 G.711 A-Law</td><td>20 ms</td></tr> <tr><td>0x01 G.711 μ-Law (Default)</td><td>20 ms</td></tr> <tr><td>0x02 G.726 16 Kbps</td><td>20 ms</td></tr> <tr><td>0x03 G.726 24 Kbps</td><td>20 ms</td></tr> <tr><td>0x04 G.726 32 Kbps</td><td>20 ms</td></tr> <tr><td>0x05 G.726 40 Kbps</td><td>20 ms</td></tr> <tr><td>0x06 G.727 16 Kbps</td><td>20 ms</td></tr> <tr><td>0x07 G.727 24-16 Kbps</td><td>20 ms</td></tr> <tr><td>0x08 G.727 24Kbps</td><td>20 ms</td></tr> <tr><td>0x09 G.727 16-32 Kbps</td><td>20 ms</td></tr> <tr><td>0x0A G.727 24-32 Kbps</td><td>20 ms</td></tr> <tr><td>0x0B G.727 32Kbps</td><td>20 ms</td></tr> <tr><td>0x0C G.727 16-40 Kbps</td><td>20 ms</td></tr> <tr><td>0x0D G.727 24-40 Kbps</td><td>20 ms</td></tr> <tr><td>0x0E G.727 32-40 Kbps</td><td>20 ms</td></tr> <tr><td>0x0F G.723 5.3 Kbps</td><td>30 ms</td></tr> <tr><td>0x10 G.723 6.3 Kbps</td><td>30 ms</td></tr> <tr><td>0x11 G.729</td><td>20 ms</td></tr> </tbody> </table>	<u>Payload Type</u>	<u>Basic Packet Rate</u>	0x00 G.711 A-Law	20 ms	0x01 G.711 μ -Law (Default)	20 ms	0x02 G.726 16 Kbps	20 ms	0x03 G.726 24 Kbps	20 ms	0x04 G.726 32 Kbps	20 ms	0x05 G.726 40 Kbps	20 ms	0x06 G.727 16 Kbps	20 ms	0x07 G.727 24-16 Kbps	20 ms	0x08 G.727 24Kbps	20 ms	0x09 G.727 16-32 Kbps	20 ms	0x0A G.727 24-32 Kbps	20 ms	0x0B G.727 32Kbps	20 ms	0x0C G.727 16-40 Kbps	20 ms	0x0D G.727 24-40 Kbps	20 ms	0x0E G.727 32-40 Kbps	20 ms	0x0F G.723 5.3 Kbps	30 ms	0x10 G.723 6.3 Kbps	30 ms	0x11 G.729	20 ms
<u>Payload Type</u>	<u>Basic Packet Rate</u>																																						
0x00 G.711 A-Law	20 ms																																						
0x01 G.711 μ -Law (Default)	20 ms																																						
0x02 G.726 16 Kbps	20 ms																																						
0x03 G.726 24 Kbps	20 ms																																						
0x04 G.726 32 Kbps	20 ms																																						
0x05 G.726 40 Kbps	20 ms																																						
0x06 G.727 16 Kbps	20 ms																																						
0x07 G.727 24-16 Kbps	20 ms																																						
0x08 G.727 24Kbps	20 ms																																						
0x09 G.727 16-32 Kbps	20 ms																																						
0x0A G.727 24-32 Kbps	20 ms																																						
0x0B G.727 32Kbps	20 ms																																						
0x0C G.727 16-40 Kbps	20 ms																																						
0x0D G.727 24-40 Kbps	20 ms																																						
0x0E G.727 32-40 Kbps	20 ms																																						
0x0F G.723 5.3 Kbps	30 ms																																						
0x10 G.723 6.3 Kbps	30 ms																																						
0x11 G.729	20 ms																																						

0x0100 RTP Payload Type (IPN Series 2)

Used in:

0x1E and 0x001E Generic PPL ICBs in:

Outsize Control message

Route Control message

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description		
Profile 1			
0, 1	Tag	0x0100 RTP Payload Type	
2, 3	Length	0x0001	
4	Value[0]	<u>Payload Type</u>	<u>Basic Packet Rate</u>
		0x00 G.711 A-Law	5 ms
		0x01 G.711 μ -Law (Default)	5 ms
Profile 2			
0, 1	Tag	0x0100 RTP Payload Type	
2, 3	Length	0x0001	
4	Value[0]	<u>Payload Type</u>	<u>Basic Packet Rate</u>
		0x00 G.711 A-Law	5 ms
		0x01 G.711 μ -Law (Default)	5 ms
		0x02 G.726 16 Kbps	5 ms
		0x03 G.726 24 Kbps	5 ms
		0x04 G.726 32 Kbps	5 ms
		0x05 G.726 40 Kbps	5 ms
		0x0F G.723 5.3 Kbps	30 ms
		0x10 G.723 6.3 Kbps	30 ms
		0x11 G.729	10 ms

0x0100 RTP Payload Type (IPN Series 3)

Used in:

0x1E and 0x001E Generic PPL ICBs in:

Route Control message

Outseize Control message

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description		
Profile 4			
0, 1	Tag	0x0100 RTP Payload Type	
2, 3	Length	0x0001	
4	Value[0]	<u>Payload Type</u>	<u>Basic Packet Rate</u>
		0x00 G.711 A-Law	5 ms
		0x01 G.711 μ -Law (Default)	5 ms
Profile 5			
0, 1	Tag	0x0100 RTP Payload Type	
2, 3	Length	0x0001	
4	Value[0]	<u>Payload Type</u>	<u>Basic Packet Rate</u>
		0x00 G.711 A-Law	5 ms
		0x01 G.711 μ -Law (Default)	5 ms
		0x02 G.726 16 Kbps	5 ms
		0x03 G.726 24 Kbps	5 ms
		0x04 G.726 32 Kbps	5 ms
		0x05 G.726 40 Kbps	5 ms
		0x0F G.723 5.3 Kbps	30 ms
		0x10 G.723 6.3 Kbps	30 ms
		0x11 G.729	10 ms

0x0101 RTP Payload Size (VDAC-ONE)

By changing the multiple of the payload size, you can configure how much encoded PCM data a packet contains.

Example: For G.711, a 160-byte payload of encoded PCM data is sent at the G.711 Basic Packet Rate of every 20 milliseconds. But if you change the multiple for G.711 to 2x, then a 320-byte payload is sent every 40 milliseconds.

In effect, changing the multiple also changes the throughput available over an Ethernet network. Larger packets create greater network delay overall, but they use the Ethernet bandwidth more efficiently. Smaller packets may use the Ethernet bandwidth less efficiently, but they may provide better voice quality.

Used in:
 0x1E Generic PPL ICB in:
Resource Attribute Query message
Resource Attribute Configure message

Byte	Description
0, 1	Tag 0x0101 RTP Payload Size
2, 3	Length 0x0001
4	Value[0] Multiple of the basic packet rate. 0x01(Default) 0x02 0x03 0x04 0x05 0x06 0x07 0x08

0x0101 RTP Payload Size (IPN Series 2)

RTP payload sizes can be changed in multiples of the specified RTP Payload Type's base packetization rate. On the IP Network Interface Series 2 card, the RTP payload size is tightly coupled with RTP payload type. Changing the default RTP payload type also changes the default RTP payload size to its applicable value (unless a new default RTP payload size is also specified). Using the non-default RTP Payload Type during call establishment results in the default RTP Payload Size not being used. If the RTP Payload Size is not provided, the default for the specific payload type is used.

In effect, changing the multiple also changes the throughput available over an Ethernet network. Larger packets create greater network delay overall, but they use the Ethernet bandwidth more efficiently. Smaller packets may use the Ethernet bandwidth less efficiently, but they may provide better voice quality.

Used in:
 0x1E Generic PPL ICB in:
Outsize Control message

0x0101 RTP Payload Size (IPN Series 3)

RTP payload sizes can be changed in multiples of the specified RTP Payload Type's base packetization rate. On the IP Network Interface Series 3 card, the RTP payload size is tightly coupled with RTP payload type. Changing the default RTP payload type also changes the default RTP payload size to its applicable value (unless a new default RTP payload size is also specified). Using the non-default RTP Payload Type during call establishment results in the default RTP Payload Size not being used. If the RTP Payload Size is not provided, the default for the specific payload type is used.

In effect, changing the multiple also changes the throughput available over an Ethernet network. Larger packets create greater network delay overall, but they use the Ethernet bandwidth more efficiently. Smaller packets may use the Ethernet bandwidth less efficiently, but they may provide better voice quality.

Used in:

0x1E Generic PPL ICB in:

Outsize Control message

Route Control message

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description	
Profile 4		
0, 1	Tag	0x0101 RTP Payload Size
2, 3	Length	0x0001
4	Value[0]	Multiple of the basic packet rate 0x04 or 0x06
Profile 5		
0, 1	Tag	0x0101 RTP Payload Size
2, 3	Length	0x0001
4	Value[0]	Multiple of the basic packet rate. Depends on the Payload Type used as follows: 0x01 - 0x03 G.723.1 (5.3 Kbps) G.723.1 (6.3Kbps) 0x02 - 0x06 (even only) - G.711 A-Law, G.711 u-Law 0x02 - 0x12 (even only)-G.726 (16, 24, 32, 40 Kbps) 0x02 - 0x06 (even only) - G.729 (8 Kbps)

0x0102 RTP Silence Suppression

The RTP Silence Suppression TLV block enables and disables silence suppression. (Do not send to an active call that is in fax/modem mode.)

Used in:

0x1E and 0x001E Generic PPL ICBs in:

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description
0, 1	Tag 0x0102 RTP Silence Suppression
2, 3	Length 0x0001
4	Value[0] 0x00 Disable (Default) 0x01 Enable

0x0103 RTP Echo Cancellation

Enables and disables Echo Cancellation and NLP.

Used in:

0x1E and 0x001E Generic PPL ICBs in:

Route Control message

Outseize Control message

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description
0, 1	Tag 0x0103 RTP Echo Cancellation
2, 3	Length 0x0001
4	Value Bit[0]:0=Master Echo Cancellation Disable 1=Master Echo Cancellation Enable (default) Bit[1]:0= NLP Enable (default) 1= NLP Disable Bits[2-7] Reserved

0x0104 RTP Packets Lost

Indicates the number of RTP packets lost.

Used in:

Channel Released with Data message

Byte	Description
0, 1	Tag 0x0104
2, 3	Length 0x0004
4-7	Value[0-3] Number of Packets Lost (4 Bytes)

0x0105 RTP Packets Received

Indicates the number of RTP packets received.

Used in:

Channel Released with Data message

Byte	Description
0, 1	Tag 0x0105
2, 3	Length 0x0004
4-7	Value[0-3] Number of Packets Received (4 Bytes)

0x0110 RTP Octets Received

Indicates the number of RTP octets received.

Used in:

Channel Released with Data message

Byte	Description
0, 1	Tag 0x0110
2, 3	Length 0x0004
4-7	Value[0-3] Number of Octets Received(4 Bytes)

0x0111 RTP Packets Sent

Indicates the number of RTP packets sent.

Used in:

Channel Released with Data message

Byte	Description
0, 1	Tag 0x0111
2, 3	Length 0x0004
4-7	Value[0-3] Number of Octets Received(4 Bytes)

0x0112 RTP Octets Sent

Indicates the number of RTP octets sent.

Used in:

Channel Released with Data message

Byte	Description
0, 1	Tag 0x0112
2, 3	Length 0x0004
4-7	Value[0-3] Number of Octets Sent (4 Bytes)

0x0115 Call Agent Mode

Enables Call Agent Mode which allows bearer-free calls.

Used in:

VoIP Protocol Configure message

Byte	Description
0, 1	Tag 0x0115
2, 3	Length 0x0001
4	Value[0] Call Agent Mode: 0x00 - Disables Call Agent Mode 0x01 - Enables Call Agent Mode

0x0116 Channel Service TLV

Indicates whether the connection being established will be switched through the CSP or will bypass the CSP.

For VoIP calls, if this TLV is not present in one of the messages listed below, a switched CSP connection is created.

For non-VoIP calls, this TLV is ignored and an CSP switched connection is created.

Used in:

0x1E Generic PPL ICB in:

PPL Event Request message

Connect with Data message

Byte	Description
0, 1	Tag 0x0116
2, 3	Length 0x0001
4, 5	Value[0-1] Bearer Service Mode: 0x0000 - Bearer-switched through the CSP which implies a physical span channel. 0x0001 - Bearer-free. The CSP is bypassed which implies a virtual span channel.

0x0117 Alerting Propagation Mode

Use this TLV to indicate whether or not to suppress the automatic generation of the alerting signal by the CSP when the connected remote party sends out an alerting signal to the CSP. This TLV is for the SIP 180 message.

Used in:

0x1E Generic PPL ICB in:

PPL Event Request message

Connect with Data message

Byte	Description
0, 1	Tag 0x0117
2, 3	Length 0x0001
4	Value[0] 1 Suppress alerting 2 Do not suppress alerting (default)

0x0118 Companding Conversion Mode

Use this TLV to specify whether PCM encoding format conversion should be bypassed or not. The default is 0 in which case PCM encoding format is not bypassed. A value of 1 bypasses the conversion process and forces the B side of the call leg to use the A side's companding format.

Used in:

0x1E and 0x001E Generic PPL ICBs in:

Route Control message

Connect with Data message

Byte	Description
0, 1	Tag 0x0118
2, 3	Length 0x0002
4, 5	Value[0] 0x0000 Companding Conversion enabled (default) 0x0001 Companding Conversion disabled

0x0119 Media Offer Stage

Use this TLV with the Alerting or Progress call processing event to ask for a media connection before the Answer message is sent.

Used in:

0x1E Generic PPL ICB in:

PPL Event Request message

Byte	Description
0, 1	Tag 0x0119
2, 3	Length 0x0002
4	Value[0] 0x0000 - Do not offer media. 0x0001 - Offer media.

0x011A Call Agent Mode Physical Channel ID

Used in:

0x1E Generic PPL ICB in:

PPL Event Request message

Connect with Data message

Byte	Description
0, 1	Tag 0x011A - Call Agent Mode Physical Channel ID
2, 3	Length 0x0003
4,5	Value[0, 1] Span ID
6	Value[2] Channel ID

0x011B Reliable Provisional Response Mode

Used in:

VoIP Protocol Configure message

Byte	Description
0, 1	Tag 0x011B
2,3	Length 0x0001
4	Value[0-3] 0 - Disabled (default) 1 - Support 2 - Require

0x011C SIP Stack T.38 Fax Support

Used in:

VoIP Protocol Configure (0x00EE)*VoIP Protocol Query* (0x00EF)

Byte	Description
0,1	Tag 0x011C SIP Stack T.38 Fax Support
2,3	Length 0x0001
4	Value[0] 00=Disabled 01=Enabled

0x0141 Modify Connection

Use this TLV to change the connection type of an established connection between two channels without having to park the channels during the transition. Add this TLV to the *Connect with Data* message along with the Connection Type TLV (0x0612). If you do not include the Connection Type TLV, the Modified Connection defaults to a two-way connection.

Used in:

0x1E Generic PPL ICB in:

Connect with Data message

Byte	Description
0, 1	Tag 0x0141 Modify Connection
2, 3	Length 0x0002
4,5	Value[0, 1] 0x0000 - Apply changes unconditionally (default)

0x0168 VDAC Module IP Address

Use this TLV to populate the Router Component (0x0064) with the VDAC module IP address.

Used in:

PPL Table Download message

Byte	Description
0, 1	Tag 0x0168
2, 3	Length 0x0004
4-7	Value[0-3] IP Address of VDAC module

0x01AE Poll Interval Configure

Use this TLV to specify the poll interval and the number of missed polls before a connection is declared down.

Used in:

VoIP Protocol Query message

VoIP Protocol Configure message

Byte	Description
0, 1	Tag 0x01AE
2, 3	Length 0x0004
4	Value[0] Poll Interval, MSB: 0x00-0xFF
5	Value[1] Poll Interval, LSB: 0x00-0xFF
6	Value[2] Number of Missed Polls Before Link is Declared Down, MSB: 0x00-0xFF
7	Value[3] Number of Missed Polls Before Link is Declared Down, LSB: 0x00-0xFF

0x01C2 Minimum Jitter Buffer Delay

The data in this TLV block are in millisecond units. The range of legitimate values is 0-150 milliseconds.

Used in:

0x1E and 0x001E Generic PPL ICBs in:

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description
0, 1	Tag 0x01C2 Minimum Jitter Buffer Delay
2, 3	Length 0x0001
4	Value[0] 0x00-0x96 ms (0x4B Default)

0x01C3 Maximum Jitter Buffer Delay (IPN-2 Card)

The data in this TLV block are in millisecond units. The range of legitimate values is 150-300 ms.

Used in:

0x1E and 0x001E Generic PPL ICBs in:

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description
0, 1	Tag 0x01C3 Maximum Jitter Buffer Delay
2, 3	Length 0x0004
4	Value[0] 0x00
5	Value[1] 0x00
6, 7	Value[2, 3] 0x0096-0x012C 0x0096 (Default)

0x01C3 Maximum Jitter Buffer Delay (IPN-2 Card)

The data in this TLV block are in millisecond units. The range of legitimate values is 150-300 ms.

Used in:

0x1E and 0x001E Generic PPL ICBs in:

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description
0, 1	Tag 0x01C3 Maximum Jitter Buffer Delay
2, 3	Length 0x0004
4	Value[0] 0x00
5	Value[1] 0x00
6, 7	Value[2, 3] 0x0096-0x012C 0x0096 (Default)

0x01C3 Maximum Jitter Buffer Delay (IPN-3 Card)

The data in this TLV block are in millisecond units. The range of legitimate values is 0-200 ms.

Used in:

0x1E and 0x001E Generic PPL ICBs in:

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description
0, 1	Tag 0x01C3 Maximum Jitter Buffer Delay
2, 3	Length 0x0004
4	Value[0] 0x00
5	Value[1] 0x00
6, 7	Value[2, 3] 0x0000-0x00C8 0x0096 (Default)

0x01C4 Adaptation Rate

Use this TLV to configure the Adaption Rate on a VDAC-ONE card.

The data in this TLV block are in millisecond units. The range of legitimate values is 0-12 milliseconds.

Used in:

0x1E and 0x001E Generic PPL ICBs in:

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description
0, 1	Tag 0x01C4 Adaptation Rate
2, 3	Length 0x0001
4	Value[0] 0x00-0x0C (0x07 Default)

0x01C5 Fax Type Enable

If the fax type is set to Disable, then the fax will go through as a voice call. Therefore, the fax transmission might not be successful.

Used in:

0x1E Generic PPL ICB in:

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description
0, 1	Tag 0x01C5 Fax Type Enable
2, 3	Length 0x0001
4	Value[0] ONE of the following: 0x00 Disable (Default) - RTP not changed. 0x01 Enable Relay 0x02 Enable Bypass

0x01C6 Modem Enable Transport Method

Used in:

0x1E Generic PPL ICB in:

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description
0, 1	Tag 0x01C6 Modem Enable Transport Method
2, 3	Length 0x0001
4	Value[0] 0x00 Transparent Mode 0x01 Modem Relay Mode (not supported) 0x02 Modem Bypass Mode

0x01C7 Fax/Modem Bypass Coder Type

Dialogic recommends using G.711 A-Law or μ -Law for fax/modem bypass mode.

Used in:

0x1E and 0x001E Generic PPL ICBs in:
Resource Attribute Query message
Resource Attribute Configure message

Byte	Description		
0, 1	Tag	0x001C7 Fax/Modem Bypass Coder Type	
2, 3	Length	0x0001	
4	Value[0]	<u>Payload Size</u>	<u>Basic Packet Rate</u>
	0x00	G.711 A-Law	20 ms
	0x01	G.711 μ -Law (Default)	20 ms
	0x02	G.726 16	20 ms
	0x03	G.726 24	20 ms
	0x04	G.726 32	20 ms
	0x05	G.726 40	20 ms
	0x06	G.727 16	20 ms
	0x07	G.727 24-16	20 ms
	0x08	G.727 24	20 ms
	0x09	G.727 32-16	20 ms
	0x0A	G.727 32-24	20 ms
	0x0B	G.727 32	20 ms
	0x0C	G.727 40-16	20 ms
	0x0D	G.727 40-24	20 ms
	0x0E	G.727 40-32	20 ms
	0x0F	G.723 5.3 kbps	30 ms
	0x10	G.723 6.3 kbps	30 ms
	0x11	G.729	20 ms

0x01C8 Protocol Type

Use this TLV to specify the protocol used by the TLVs in the *VoIP Protocol Configure* and *VoIP Protocol Query* messages.

Used in:

0x1E Generic PPL ICB in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description	
0, 1	Tag	0x01C8
2, 3	Length	0x0001
4	Value[0]	0x04 SIP 0x05 H.323 0x06 RFC 2833

0x01C9 Primary Matrix IP Port

Use this TLV to specify the IP address and port number for the primary Call Agent. The port number must be greater than 0x0400 (1024).

Used in:

VoIP Protocol Query message

VoIP Protocol Configure message

Byte	Description
0, 1	Tag 0x01C9
2, 3	Length 0x0006
4-7	Value[0-3] IP Address
8, 9	Value[1, 2] Port Number

0x01CA Secondary Matrix IP Port

Use this TLV to specify the IP address and port number for the secondary Call Agent. The port number must be greater than 0x0400 (1024).

Used in:

VoIP Protocol Query message

VoIP Protocol Configure message

Byte	Description
0, 1	Tag 0x01CA
2, 3	Length 0x0006
4-7	Value[0-3] IP Address (4 bytes)
8, 9	Value[1, 2] Port Number

0x01D0 Gateway Mode

Use this TLV to configure an IP Network Interface Series 2 card to be in Gateway Mode or Normal Mode.

Important! This TLV is addressed to a card, and therefore cannot be combined in a message with any other TLVs.

Used in:

Resource Attribute Configure message

Byte	Description
0, 1	Tag: Gateway Mode (0x01D0)
2, 3	Length: 0x0001
4	Value: Setting 0x01 Gateway Mode (Non-port Consuming Mode) 0x02 Normal Mode (Port Consuming Mode)

0x01D1 RTP Payload Redundancy

Use this TLV to set the redundancy packet depth for RTP at configuration time. You can also set the RTP packet depth in real time, using an *Outseize Control* or *Route Control* message.

Used in:

0x1E Generic PPL ICB in:

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description
0, 1	Tag: 0x01D1 RTP Payload Redundancy
2, 3	Length: 0x0001
4	Value[0] 0x00 No Redundancy (Default) 0x01 Redundancy Level 1 (Not supported by Profile 4 on the IPN-3 card.)

0x01D2 Fax Payload Redundancy

Use this TLV to set the redundancy packet depth for Fax at configuration time only.

Used in:

0x1E Generic PPL ICB in:

Resource Attribute Query message

Resource Attribute Configure message

Byte	Description
0, 1	Tag: 0x01D2 Fax Packet Depth
2, 3	Length: 0x0001
4	Value[0] ONE of the following: 0x00 No Redundancy (Default) 0x01 Redundancy Level 1 0x02 Redundancy Level 2 (VDAC-ONE card maximum) 0x03 Redundancy Level 3 (IP Network Interface Series 3 card only)

0x01D4 Type of Service

Use this TLV to set the quality of service for outgoing packets (RTP/RTCP/T.38). This attribute can be modified in a run-time environment.

Note that the signaling protocol is responsible to adhere to the RFC Standards when imposing type of service restrictions.

Used in:

0x1E Generic PPL ICB in:

Resource Attribute Configure message

Resource Attribute Query message

Byte	Description
0, 1	Tag: Type of Service (0x01D4)
2, 3	Length: 0x0001
4	Value: Data[0] = b ₇ b ₆ b ₅ b ₄ b ₃ b ₂ b ₁ b ₀ b ₇ b ₆ b ₅ : Precedence: 000 Routine 100 Priority 010 Immediate 110 Flash 001 Flash Override 101 CRITIC/ECP 011 Internetwork control 111 Network control 0x00 (Default) b ₄ : Delay: 0 Normal delay (Default) 1 Low delay b ₃ : Throughput: 0 Normal throughput (Default) 1 High throughput b ₂ : Reliability: 0 Normal reliability (Default) 1 High reliability b ₁ : Cost: 0 Normal cost (Default) 1 Low cost b ₀ : This bit must be zero

0x01D6 DNS IP Address

Use this TLV to configure the DNS lookup in the CSP Matrix Series 3 Card. Use 0x00000000 if the Secondary or Tertiary DNS IP address is not used.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x01D6
2, 3	Length 0x000C
4-7	Value[0-3] Primary DNS IP
8-11	Value[4-7] Secondary DNS IP
12-15	Value [8-11] Tertiary DNS IP

0x01D7 Default Domain Name

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x01D7
2, 3	Length Variable
:	Value Null-terminated ASCII string for default domain

0x01DB Connection Mode

Use this TLV to set the connection mode. A line can be connected to receive only, transmit only, or both.

Used in:

Resource Attribute Configure message

Resource Attribute Query message

Byte	Description
0, 1	Tag Connection Mode (0x01DB)
2, 3	Length 0x0001
4	Value[0] 0x00 Two-way connection 0x01 One-way listen/receive only 0x02 One-way talk/transmit only 0x03 Hold connection. No voice is transmitted or received. 0xFF (Default) You see this value only if you query the default settings of a VoIP module or when the call is not connected.

0x01DC VoIP Module

Use this TLV to indicate a VoIP module.

Used in:
 0x00FB *ARP Cache Report* message

Byte	Description
0, 1	Tag 0x01DC VoIP Module
2, 3	Length 0x0001
4	Value[0] Module Number

0x01DD Flush ARP Cache Table Entry

Send this TLV to remove entries from an ARP Cache table. You must also include a VoIP Module TLV to indicate the module.

To remove all entries in the table, enter a length of 0x0000.

To remove an entry for a specific IP address, enter a length of 0x0004 and enter the IP address in Data[4-7].

To remove an entry for a specific Ethernet address, enter a length of 0x0006 and enter the Ethernet address in Data[4-9].

Used in:
ARP Cache Query

Byte	Description
0,1	Tag Flush ARP Cache Table Entry (0x01DD)
2, 3	Length Variable (see below)
4	Value:0x0000 To remove all entries on the module. There is no data. 0x0004 To remove a specific IP address. Also enter the IP address in Data[4-7]. 0x0006 To remove all entries at an Ethernet Address. Also enter the Ethernet address in Data[4-9].

0x01DE ARP Cache Table Entry

This TLV is used in the *ARP Cache Report* message to report the IP Address and Ethernet Address for an entry in the ARP Cache table. Separate TLVs are sent for each entry in the table.

Byte	Description
0, 1	Tag 0x01DE ARP Cache Table Entry
2, 3	Length 0x000E
4-7	Value[0-3] Entry Number
8-11	Value[4-7] IP Address
12-17	Value[8-13] Ethernet Address

0x01DF UDP Source Port Validate

Use this TLV to enable a VoIP media stream to be checked for its validity by authenticating the UDP source port of the incoming packets. The default value is 1 and implies that the UDP source port validation is performed on the received packets. By setting the value of this TLV to 1, the UDP source port field in every received packet (whether RTP, RTCP or T.38 packet) is validated.

Used in:

0x1E Generic PPL ICB in:

Resource Attribute Configure message

Resource Attribute Query message

Byte	Description
0,1	Tag 0x01DF (UDP Source Port Validate)
2,3	Length 0x0001
4	Value[0] ONE of the following: 0x00 Disable 0x01 Enable (Default)

0x01E0 ARP Cache Report Information

This TLV is sent in the *ARP Cache Report* message to indicate the number of messages required for the report and the number of the current message.

Byte	Description
0	Tag 0x01E0
1	Length 0x0002
2	Value[0] Number of <i>ARP Cache Report</i> messages
3	Value[1] Current Message Number

0x01E1 Fax Compatibility Mode

Use this TLV to determine the VDAC-ONE card's T.38 Fax Relay Compatibility Mode:

- Backward Compatible mode

or

- Interoperability mode.

These modes are incompatible. You can choose one mode or the other, but not both. The Backward Compatible mode is interoperable with the fax relay mode on the previously-released software. The Interoperability Mode is primarily used to interoperate with third party T.38 fax relay.

This TLV can be configured only before the call is established. It cannot be configured during an active call. This TLV does not apply to the IP Network Interface Series 2 card.

Used in:

0x1E Generic PPL ICB in:

Resource Attribute Configure message

Route Control message

Outseize Control message

Byte	Description
0, 1	Tag 0x01E1
2, 3	Length 0x0001
4	Value[0] 0x00 Backward compatible mode (Default) 0x01 Interoperability mode

0x01E2 RFC 2833 Enable

Use this TLV to allow the in-band DTMF signals to be relayed within the VoIP media stream utilizing a special RTP payload type. Low bit-rate audio codecs (such as G.729 or G.723.1) can compromise the signal integrity of DTMF digits (and other telephony tones and signals); hence causing inaccurate and poor detection/recognition of the DTMF digits at the recipient side. When this feature is enabled the incoming DTMF digits, if detected, are removed from the audio stream by the VDAC-ONE or IP Network Interface Series 2 card connected to the originator of the DTMF digits. The information regarding the detected and removed DTMF digits is embedded within the RTP stream using a special payload format. The VDAC-ONE or IP Network Interface

Series 2 card at the receiving side will decode this special RTP payload that carries DTMF digits and regenerate the DTMF signal toward the receiving PSTN side.

If you are using the VDAC-ONE card, this TLV can be configured only before or during the call establishment. It cannot be set during an active call. If you are using the IP Networking Series 2 card, this TLV can be configured while the call is active.

Important! The CSP does NOT support RFC 2833 via H.323 signaling.

Used in:

0x1E Generic PPL ICB in:

Resource Attribute Configure/Query message

Route Control message

Outseize Control message

Byte	Description
0, 1	Tag 0x01E2
2, 3	Length 0x0001
4	Value[0] 0x00 RFC 2833 disabled (in-band digit transmission) (Default) 0x01 RFC 2833 enabled (out-of-band digit transmission) 0x02 Digit transmission disabled

0x01E5 VoIP Resource Profile Assign

The Resource Profile Assign (0x01E5) TLV has been created to configure a module's resource profile. Similar to changing the Gateway Mode, changing a VoIP module resource profile causes its TDM resources to change, causing the IP Network Interface Series 2 card to reset, so that new resources can be allocated.

Important! Changing the resource profile on any module causes the IP Network Interface Series 2 card host configuration to be reset.

You cannot mix VoIP resource profiles on the same IP Network Interface Series 2 card or in the same chassis.

The Resource Profile Assign (0x01E5) TLV addresses the VoIP module by its module number. Since updating a VoIP module's profile causes the IP Network Interface Series 2 card to reset, all of the modules can be reset in a single message.

The host has the flexibility to send one VoIP Resource Profile Assign TLV to assign the same resource profile to all modules or send multiple TLVs to individual modules. If any TLV is invalid, the message is negatively acknowledged (NACKED) and no action is taken.

Used in:

Resource Attribute Configure message

Byte	Description
0, 1	Tag VoIP Resource Profile Assign (0x01E5)
2, 3	Length 0x0003
4	Value: Data [0] Module Number 0x00 Module 1 0x01 Module 2 0xFF All Populated VoIP Modules
5	Value: Data [1] Profile Number 0x01 Profile 1 (IP Network Interface Series 2 - G.711 Only) 0x02 Profile 2 (IP Network Interface Series 2 - VDAC-ONE Compatibility.) 0x04 Profile 4 (IP Network Interface Series 3 - G.711 Only) 0x05 Profile 5 (IP Network Interface Series 3 - LBR+T.38)
6	Value: Data [2] Force Flag 0x00 Update only if Spans have not been configured. 0xFF Update Unconditionally

0x01E6 Source T.38 Port

Use this TLV to specify the local T.38 port number for the fax relay. This TLV can be set on-the-fly. There is no default value for this TLV (i.e., no default T.38 port will be assumed). Therefore is imperative for the Host to resolve this port number as early as possible to ensure the proper fax relay operation. If this port number is not specified (in time), the fax relay will fail.

Used in:

0x1E Generic PPL ICB in:

Resource Attribute Configure message

Route Control message

Outseize Control message

Byte	Description
0, 1	Tag 0x01E6
2, 3	Length 0x0002
4, 5	Value[0, 1] T.38 Port

0x01E7 Destination T.38 Port

Use this TLV to specify the remote T.38 port number for the fax relay. This TLV can be set on-the-fly. There is no default value for this TLV (i.e., no default T.38 port will be assumed). Therefore is imperative for the Host to resolve this port number as early as possible to ensure the proper fax relay operation. If this port number is not specified (in time), the fax relay will fail.

Used in:

0x1E Generic PPL ICB in:

Resource Attribute Configure message

Route Control message

Outseize Control message

Byte	Description
0, 1	Tag 0x01E7
2, 3	Length 0x0002
4, 5	Value[0, 1] T.38 Port

0x01E8 Source RTCP Port

Use this TLV to specify the local RTCP port number for the fax relay. This TLV can be set on-the-fly. If this port is not specified by the host during the call setup, per RFC 1889, by default the source RTCP port is at an offset of 1 from the source RTP port.

Used in:

0x1E Generic PPL ICB in:

Resource Attribute Configure message

Route Control message

Outseize Control message

Byte	Description
0, 1	Tag 0x01E8
2, 3	Length 0x0002
4, 5	Value[0, 1] RTCP Port

0x01E9 Destination RTCP Port

Use this TLV to specify the remote RTCP port number for the fax relay. This TLV can be set on-the-fly. If this port is not specified by the host during the call setup, per RFC 1889, by default the destination RTCP port is at an offset of 1 from the destination RTP port.

Used in:
 0x1E Generic PPL ICB in:
Resource Attribute Configure message
Route Control message
Outseize Control message

Byte	Description
0, 1	Tag 0x01E9
2, 3	Length 0x0002
4, 5	Value[0] RTCP Port

0x01EA VoIP Terminal Capabilities

The *Resource Attribute Query* message is used to query an IP endpoint’s terminal capabilities using this TLV. The report returned is based on the resource profile assigned to the module.

Important! The VoIP Terminal Capabilities can be queried by addressing either the IP Address AIB or Extended Span/Channel AIB embedded within the Address Element TLV.

Used in:
Resource Attribute Query message

Byte	Description
0, 1	Tag VoIP Resource Profile Assign (0x01EA)
2, 3	Length Variable
4	Data [0] VoIP Profile ID 0x00 VDAC-ONE 0x01 IP Network Interface Series 2 - G.711 Only 0x02 IP Network Interface Series 2 - LBR + T.38 0x03 Reserved
5	Data [1] Echo Canceler Tail Length (in ms)
6, 7	Data [2] MSB Jitter Buffer Length (in ms) Data [3] LSB Jitter Buffer Length (in ms)
8	Data [4] Silent Suppression Support (bitmask) 0x00 No Support 0x01 Generic VDAC Support 0x02 G.723.1 Annex A Support 0x03 G.729 Annex B Support
9	Data [5] Fax Relay Supported (bitmask) 0x00 No Support 0x01 Support using T.38
11	Data [6] Modem Relay Supported (bitmask) 0x00 No Support

12	Data [7] Digit Relay Support (bitmask) 0x00 No Support 0x01 Support using RFC 2833 0x03 Support using RFC 2833 and Payload Type Dynamically Configurable (IP Network Interface Series 2 only)
13	Data [8] RTP Redundancy Support (bitmask) 0x00 No Support 0x01 Support using RFC 2198 0x03 Support using RFC 2198 and Payload Type Dynamically Configurable (IP Network Interface Series 2 only)
14	Data [9] Number of Coders Supported (bitmask)
15	Vocoder [1] Payload Type
16	Vocoder [1] Basic Packet Rate (in ms)
17	Vocoder [1] Maximum Packet Rate (multiples of Basic Rate)
...	...
...	Vocoder [n] Payload Type
...	Vocoder [n] Basic Packet Rate (in ms)
14+n*3	Vocoder [n] Maximum Packet Rate (multiples of Basic Rate)

0x01EB RTP Timer Timeout (VDAC-ONE)

Allows you to create a timer for monitoring the first incoming packet (for example, RTP, RTCP, or T.38) for each channel. If the timer expires before the first packet is received, a PPL Event Indication is generated, with an Event ID of 0x07.

Used in:

0x1E Generic PPL ICB in:

Resource Attribute Configure message

Route Control message

Outseize Control message

Byte	Description
0, 1	Tag 0x01EB
2, 3	Length 0x0004
4, 5	Value[0, 1] 0x0000
6, 7	RTP Timer Timeout Value (2 Bytes - 10 ms increments) 0x0000 - 0xFFFF ms 0x0000 is the default, and it means that the timer is disabled. Any other value automatically enables the timer to that value.

0x01EB Initial Media Inactivity Timeout (IP Network Interface Series 2 Card)

Allows you to create a timer for monitoring the first incoming packet (for example, RTP, RTCP, or T.38) for each channel. If the timer expires before the first packet is received, a PPL Event Indication is generated, with an Event ID of 0x07.

Used in:

0x1E Generic PPL ICB in:

Resource Attribute Configure message

Route Control message

Outseize Control message

Byte	Description
0, 1	Tag 0x01EB
2, 3	Length 0x0004
4, 5	Value[0, 1] 0x0000
6, 7	RTP Timer Timeout Value (2 Bytes - 10 ms increments) 0x0000 - 0xFFFF ms 0x0000 is the default, and it means that the timer is disabled. Any other value automatically enables the timer to that value.

0x01EC Media Inactivity Detection Timer

Use this TLV to create a timer for monitoring the incoming RTCP packets (VDAC-ONE) or UPD packets (IPN Series 2) at a per module or a per channel basis to detect possible media inactivity occurrence. When this timer is enabled for a particular channel, if the configured timer value has elapsed and no RTCP/UPD packets are received, then this timer will expire and generate a PPL Event Indication with Event ID of 0x08 to alert the host of a possible media inactivity detection. RTCP/UPD must be supported by remote endpoints for this TLV to work.

Used in:

0x1E Generic PPL ICB in:

Resource Attribute Configure message

Route Control message

Outseize Control message (not for SIP or H.323)

Byte	Description
0, 1	Tag 0x01EC
2, 3	Length 0x0004
4, 5	Value[0,1] 0x0000
6, 7	RTP Timer Timeout Value (2 Bytes - 10 ms increments) 0x00 - 0xFFFF ms. 0x00 is the default, and it means that the timer is disabled. Any other value automatically enables the timer to that value.

0x01ED RFC 2833 Jitter Buffer Size

Indicates the size of the RTP jitter buffer in milliseconds. The range is 0-400 milliseconds. The default is 200 milliseconds.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x01ED
2, 3	Length 0x0002
4, 5	Value[0] Value in milliseconds. 0x0000-0x0190 (0-400 ms) 0x00C8 (200 ms) is the default

0x01EE Invalid Packet Alarm Threshold

Indicates the alarming threshold for the number of invalid non-RFC 2833 packets received by the CSP. The CSP generates an alarm every time the number of invalid packets exceeds the threshold limit in a one-second period. The alarming is on a per RFC 2833 channel basis.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x01EE
2, 3	Length 0x0002
4, 5	Value[0,1] Number of invalid packets that justify alarm generation. 0x0000 - 0xFFFF 0x0032 is the default

0x01EF Initial UDP Port Number

Specifies the first UDP port number that will be used for receiving the RFC 2833 stream from an endpoint. The CSP increments the port number beyond this "seed" by two for every new call. The port number rolls over to the initial value once all the configured ports are exhausted.

You specify the number of UDP ports in the Number of UDP Ports 0x01F0.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x01EF
2, 3	Length 0x0002
4, 5	Value[0,1] Initial port number above 1024. 0x0400 - 0x2710 (1024-10,000) 0x2710 is the default

0x01F0 Number of UDP Ports

Specifies the total number of RFC 2833/RTP/UDP ports that the CSP is allowed to open. For a multiport scheme, this number is the number of RFC 2833 channels requested by a host application. Select 0 to specify that only one port needs to be opened that will multiplex all of the RFC 2833 streams. Any value other than 0 implies a multi-port scheme and the actual number of UDP ports required for each RFC 2833 channel basis.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x01F0
2, 3	Length 0x0002
4, 5	Value[0,1] 0 - implies single port scheme. 1 is the default Any other value implies multiport scheme. The number should be equal to or less than the the CSP maximum channel capacity.

0x01F1 RFC 2833 Dynamic Payload Type

Indicates the dynamic payload type number that the CSP will utilize for specifying the RFC 2833 multi-unicast media stream to the peer endpoint.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Outsize Control message

Route Control message

Resource Attribute Configure message

Byte	Description
0, 1	Tag 0x01F1
2, 3	Length 0x0001
4, 5	Value[0,1] 96-127 Dynamic Payload Type (0x60 - 0x7F) Default is 96 (0x60) Values 102, 103, and 105 (0x66, 0x67, 0x69) are reserved. The VDAC-ONE card rejects these values. The IPN-2 card allows using these values but the host application is responsible for resolving conflicts.

0x01F2 RFC 2198 Dynamic Payload Type

Indicates the dynamic payload type number that the CSP will utilize for specifying the RFC 2198 redundant RTP payload to the peer end-point.

Used in:

VoIP Protocol Configure message

Outseize Control message

Route Control message

Resource Attribute Configure message

Byte	Description
0, 1	Tag 0x01F2
2, 3	Length 0x0001
4, 5	Value[0,1] 96-127 Dynamic Payload Type (0x96 - 0x7F) Default is 104 (0x68) Values 102, 103, and 105 (0x66, 0x67, 0x69) are reserved. The VDAC-ONE card rejects these values. The IPN-2 card allows using these values but the host application is responsible for resolving conflicts.

0x01F4 Media Recovery Method

Defines how a module on an IPN-2 card recovers from DSP failures that result in the loss of VoIP channels.

Used in:

Resource Attribute Configure message

Resource Attribute Query message

Byte	Description
0, 1	Tag 0x01F4
2, 3	Length 0x0004
4	<p>Value[0] Recovery Procedure</p> <p>0x00 - Disable (Default) No special recovery options; backward compatible</p> <p>0x01 - Immediate forced bleed-off</p> <p>Immediate forced bleed-off of all resources from the Calisto DSP.</p> <p>Calisto DSP is reset when there are no active resources.</p> <p>All four Calisto DSPs in the module gets reset one after the other.</p> <p>This recovery procedure is not cyclic. It runs once when this method is enabled by host. After the procedure completes the media recovery method shall be disabled.</p> <p>0x02 - Immediate forced bleed-off on loss of DSP resources</p> <p>Immediate forced bleed-off of all resources in a Calisto DSP when the number of lost resources in a DSP goes below the threshold values.</p> <p>Calisto DSP reset when no active resources.</p> <p>Only that single Calisto DSP is reset.</p> <p>This recovery procedure is cyclic. It will remain enabled until the it is disabled explicitly. The procedure is triggered whenever the fatal error exceeds the threshold value.</p> <p>The fatal error count shall be reset after the procedure completes.</p>

5	<p>Value[1] Max wait time</p> <p>The maximum time to wait for all resources to become idle in the calisto DSP chips. On timeout the active calls in the Calisto DSP are dropped. This field is counted in units of 20 seconds.</p> <p>0x00 – Do not wait reset immediately (Forced Reset)</p> <p>0x01 to 0xFE – Wait for Value[1]*20 seconds (20s to 1h 24m 40s)</p> <p>0xFF – Wait forever till all the channels are idle</p> <p>This field is not applicable if Value [0] is set to 0x00.</p>
6	<p>Value[2] DSP fatal error threshold value</p> <p>The number of fatal errors to happen in a single Calisto to before triggering the media recovery procedure.</p> <p>Valid Values range from 1 to 127 for Profile 1</p> <p>Valid Values range from 1 to 62 for Profile 2</p>
7	<p>Value[3] Calisto (DSP) Number to be reset</p> <p>This field is a bit map of DSPs to be reset.</p> <p>0x00 – Reset all DSPs</p> <p>Bit 0 – Set if DSP Number 0 is to be reset</p> <p>Bit 1 – Set if DSP Number 1 is to be reset</p> <p>..</p> <p>Bit n – Set if DSP Number n is to be reset</p> <p>The valid Calisto numbers range from 0x00-0x007.</p> <p>This field is applicable only if Value[0] is set to 0x01.</p>

0x0262 Use SIP Local Port

Use this TLV to specify the local SIP port number. The default port number is 0x13C4 (5060).

Used in:

VoIP Protocol Configure message

Byte	Description
0, 1	Tag 0x0262
2, 3	Length 0x0002
4, 5	Value[0,1] Port Number

0x0263 Use SIP Remote Port/Default Proxy Port

Use this TLV to specify the port number of a remote SIP server. The default port number is 0x13C4 (5060).

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x0263
2, 3	Length 0x0002
4, 5	Value[0,1] Port Number 5060 (default)

0x0264 Use SIP Remote Host/Default Proxy Host

Use this TLV to specify the remote default proxy host. The value is a null-terminated ASCII string representing either an IP address (for example, 10.10.10.10) or domain name (for example, sip.excel.com). To de-configure the SIP default proxy host use this value: 0.0.0.0.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x0264
2, 3	Length Variable
4-N	Value Null terminated ASCII string

0x0265 Use SIP Site ID

Use this TLV to specify the site ID that is used to generate the call ID. The default value is UNSET.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x0265
2, 3	Length Variable
4 . . .	Value[0 . . .] Site ID (Null terminated ASCII text string) EXCEL (Default)

0x0266 Use SIP Anonymous Caller

Use this TLV to specify the default caller number that is used when no caller number is specified. The default caller number is 0x0000000000000000 (00000000).

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x0266
2, 3	Length Variable
4 . . .	Value[0 . . .] Caller Number (Null terminated ASCII text string)

0x0267 Use SIP Invite T1 Timer

Use this TLV to specify the T1 timer value for an INVITE request in milliseconds. The default value is 0x1F4 (500 milliseconds).

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x0267
2, 3	Length 0x0004
4-7	Value[0-3] T1 Timer (4 bytes)

0x0268 Use SIP BYE T1 Timer

Use this TLV to specify the T1 timer value for BYE in milliseconds. The default value is 0x1F4 (500 milliseconds).

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x0268
2, 3	Length 0x0004
4-7	Value[0-3] T1 Timer (4 Bytes)

0x0269 Use SIP T2 Timer

Use this TLV to specify the T2 timer value in milliseconds. The default value is 0xFA0 (4000 milliseconds).

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x0269
2, 3	Length 0x0004
4-7	Value[0-3] T2 Timer (4 Bytes)

0x026A Use SIP Maximum Retransmissions Invite

Use this TLV to specify the maximum retransmissions for an INVITE request. The default value is 0x07.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x026A
2, 3	Length 0x0001
4	Value[0] Maximum Number of Transmissions

0x026B Use SIP Max Retransmissions BYE

Use this TLV to specify the maximum retransmissions for a BYE request (any request other than INVITE). The default value is 0x0B.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x026B
2, 3	Length 0x0001
4	Value[0] Maximum Transmissions

0x0270 SIP Tunnel Type

For all outgoing messages, this TLV enables and disables the use of tunneled data and specifies the tunneled data type.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x0270
2, 3	Length 0x0002
4-6	Value[0-2] 0x0001 No tunneling (Default) 0x0002 CSP UPDF Tunneling 0x0004 Customize MIME Body

0x0272 Use SIP Registration Timeout

Use this TLV to specify the default registration expiration period in seconds that is used when no registration expiration period is specified. The default value is 0x00000E10 (3600 seconds).

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x0272
2, 3	Length 0x0004
4-7	Value[0-3] Registration Expiration Period (4 Bytes)

0x0274 Use No Local Media

Use this TLV to specify the way to handle a situation where there is no VDAC-ONE or IP Network Interface Series 2 card.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x0274
2, 3	Length 0x0001
4	Value[0] Media Handling 0x00 Not in the media path - no VDAC-ONE or IP Network Interface Series 2 cards available 0x01 Terminate media locally - VDAC-ONE or IP Network Interface Series 2 cards (Default)

0x0275 Local Registration Enable/Disable

Use this TLV to enable or disable the acceptance of local registrations within the CSP.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x0275
2, 3	Length 0x0001
4	Value[0] Local Registration Enable/Disable 0x00 Disable local storage of registrations 0x01 (Default) Store up to 2048 SIP phone registrations locally in the CSP.

0x0277 SIP Version

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x0277
2, 3	Length 0x0002
4,5	Value[0,1] MSB - Major Version LSB - Minor Version

0x0279 SIP Local Route Lookup

Use this TLV to enable and disable local registration lookup, even if the Route Control message has the “To” IP address. If this TLV is enabled, the supplied “To” IP address is ignored. If you use internal routing and you want to make a SIP-to-SIP call, Dialogic recommends enabling this TLV. You can enable and disable the local route lookup feature for an individual call by using the NPDI SIP Local Lookup TLV (0x2914).

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x0279
2, 3	Length 0x0001
4	Value[0] 0x00 Disable (Default) 0x01 Enable

0x027A SIP Registration Mode

Use this TLV to configure the CSP for different registration modes.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description								
0, 1	Tag 0x027A								
2, 3	Length 0x0001								
4	Value[0] <table border="0" style="margin-left: 20px;"> <tr> <td>0x00</td> <td>Do not notify host (Default) All registrations are handled locally, without host intervention.</td> </tr> <tr> <td>0x01</td> <td>Notify host Registrations are stored locally, and the host is notified by a PPL Event Indication. The host has no control over the response generated.</td> </tr> <tr> <td>0x02</td> <td>Notify host and wait for response All registrations are added to the local database. The host is notified by a PPL Event Indication, and waits for a PPL Event Request before sending a response to the UA client.</td> </tr> <tr> <td>0x03</td> <td>Accept no registration</td> </tr> </table>	0x00	Do not notify host (Default) All registrations are handled locally, without host intervention.	0x01	Notify host Registrations are stored locally, and the host is notified by a PPL Event Indication. The host has no control over the response generated.	0x02	Notify host and wait for response All registrations are added to the local database. The host is notified by a PPL Event Indication, and waits for a PPL Event Request before sending a response to the UA client.	0x03	Accept no registration
0x00	Do not notify host (Default) All registrations are handled locally, without host intervention.								
0x01	Notify host Registrations are stored locally, and the host is notified by a PPL Event Indication. The host has no control over the response generated.								
0x02	Notify host and wait for response All registrations are added to the local database. The host is notified by a PPL Event Indication, and waits for a PPL Event Request before sending a response to the UA client.								
0x03	Accept no registration								

0x027B Session Interval

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Use this TLV to set the value of the session-expire header (in seconds) in the SIP messages generated by the CSP.

Byte	Description
0, 1	Tag 0x027B
2, 3	Length 0x0004
4-7	Value[0] Variable 1800 (30 minutes) (Default) 0x00 00 07 08

0x027C Min-SE Interval

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Use this TLV to set the Min-SE header field (in seconds) which conveys the minimum allowed value for the session timer. The Min-SE Interval has to be less than the Session Interval. The maximum value is 0x7F FF FF FE.

Byte	Description
0, 1	Tag 0x027C
2, 3	Length 0x0004
4-7	Value[0] Variable 300 (5 minutes) (Default) 0x00 00 01 2C

0x027D SIP Transport Type

Specifies the default transport type for outbound call. If this TLV does not appear in the VoIP Protocol Configure message, the CSP uses the default - UDP.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x027D
2, 3	Length 0x0001
4-7	Value[0-3] 1 - UDP 2 - TCP

0x027F SIP Message Information Mask

Used in:

VoIP Protocol Configure message*VoIP Protocol Query* message

The host uses this TLV to specify the additional SIP message information to the host in the *Request for Service with Data* message.

Byte	Description
0, 1	Tag 0x027F
2, 3	Length 0x0004
4-7	<p>Value[0-3] This field is a 32-bit mask. Each bit selects specific SIP message fields and is listed from LSB to MSB.</p> <p>0 - Disabled (Default) 1 - Enabled</p> <p>Bit 0 Selects the Dialog Information that contains Call-ID, From Tag and To Tag</p> <p>Bit 1 Selects the Proxy-Authorization Header</p> <p>Bit 2 Selects the Authorization Header</p> <p>Bit 3 Selects the Request for URI information</p> <p>Bit 4 Selects Media Connection Address</p> <p>Bit 5 Selects the Contact URI Parameter information</p> <p>Bit 6 Selects the Request URI Parameter information for proprietary information</p> <p>Bit 7 Selects reporting of Remote Party ID header field(s)</p> <p>Bit 8 Selects reporting of RPID Privacy header field(s)</p> <p>Bit 9 Selects reporting of Subject header field</p> <p>Bit 10 Selects reporting of Via header field</p> <p>Bit 11 Selects reporting of Subject header field in the REFER message.</p> <p>Bit 12 Selects reporting of Refer-To Header Parameters.</p> <p>Bit 13 Selects reporting of Contact display name, username and parameter in 200OK for INVITE and REFER</p> <p>Bit 14 Selects reporting of SIP Supported and Require option tags in the INVITE, 180, and 183 messages</p> <p>Bit 15 Selects reporting of P-Headers and Privacy header</p> <p>Bit 16 Selects reporting of Call-ID for SIP outbound calls.</p> <p>Bit 17 Selects SDP Non-Audio Media Stream Native Text Propagation in Call Agent Mode</p> <p>Bit 18 - Selects reporting of Referred-By header in REFER and INVITE request</p> <p>Bit 19 - Selects reporting Subscription Stack header</p> <p>Bit 20 - Selects reporting of MIME data in incoming messages.</p> <p>Bit 21 - Selects reporting Refer Request URI</p> <p>Bit 22 - Selects reporting SIP Access to Parameter in To Header</p> <p>Bits 23-31 Reserved</p>

0x027E SIP Persistent Sockets

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

This TLV forces all TCP sockets to be persistent.

Byte	Description
0, 1	Tag 0x027E
2, 3	Length 0x0001
4-7	Value[0-3] 0 disable (default) 1 enable

0x0280 SIP Existing Socket Reuse

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

For Connection oriented sockets, this TLV allows (or prevents) the reuse of any SIP socket which may exist between this host and the target endpoint. When this TLV is disabled, a new socket will be opened for each SIP Request.

Byte	Description
0, 1	Tag 0x0280
2, 3	Length 0x0001
4-7	Value[0-3] 0 disable (default) 1 enable

0x0281 SIP Idle Socket Timeout

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

For connection oriented sockets that are not persistent, this TLV maintains an idle timer to determine when a socket should be closed.

Byte	Description
0, 1	Tag 0x0281
2, 3	Length 0x0002
4-7	Value[0-3] The number of idle seconds to wait before closing a non-persistent socket. Default is 32 seconds. Minimum is five seconds. Maximum is 32,766 seconds.

0x0282 PPL Event Notification Mask

Used in:

VoIP Protocol Configure message*VoIP Protocol Query* message

Byte	Description
0, 1	Tag 0x0282
2, 3	Length 0x0004
4-7	<p>Value[0-3] 32 bit mask with each bit selects specific PPL Event Indication.</p> <p>Bit 0 - Media changed via the RE-INVITE message (Event ID 0x001E)</p> <p>Bit 1 - 200 OK Received (Event ID 0x0020)</p> <p>Bit 2 - 183 received (Event ID 0x001F)</p> <p>Bit 3 - 180 received (Event ID 0x0024)</p> <p>Bit 4 - ACK with SDP (Delayed Media)</p> <p>Bit 5 - PRACK received (Event ID 0x002A)</p> <p>Bit 6 - On server side, report INFO message received (Event ID 0x002C). On client side, report the response received for INFO message sent (Event ID 0x002D).</p> <p>Bit 7 - 182 Received (Event ID 0x002E)</p> <p>Bit 8 - Options message received. (Event ID 0x002F)</p> <p>Bit 9 - 181 Received (Event ID 0x0030)</p> <p>Bit 10 - 422 Received (Event ID 0x0031)</p> <p>Bits 8-31. Undefined.</p> <p>Bits 0, 1, 2, 4, 5, and 10 are enabled by default. All other bits are disabled by default.</p>

0x0283 Minimum Registration Duration

Used in:

VoIP Protocol Configure message

Byte	Description
0, 1	Tag 0x0283 - Minimum Registration Duration
2,3	Length 0x0004
4-7	Value[0-3] Time period in seconds

0x0284 Resynchronize VoIP Media Parameter

Use this TLV to resynchronize VoIP media parameters (maintained locally by SIP and H.323 software) after the *Resource Attribute Configure* message is sent to the CSP to change the underlying VoIP media resource configuration.

Used in:

VoIP Protocol Configure message

Byte	Description
0, 1	Tag 0x0284
2, 3	Length 0x0001
4-7	Value[0-3] 0 - No operation 1 - Resynchronize

0x0286 SIP Max Forwards

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x0286
2, 3	Length 0x0002
4,5	Value[0] 0x00 - Disable 0x01 - Enable
:	Value[1] Variable (0x00-0xFF) Max-Forwards value

0x02BC Gatekeeper IP and Port

The Gatekeeper IP and Port Configure TLV is used only if the RAS module is configured during the stack configuration.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x02BC Gatekeeper IP and Port
2, 3	Length 0xNNNN
4	Value[0] Configure/Deconfigure option 0x00 Configure (Default) 0x01 Deconfigure
5	Value[1] Number of Gatekeepers. 0x01 (Default) First entry is primary Gatekeeper. The rest are alternate Gatekeepers 0xM (M = 0x01 - 0x06)
6-9	Value[2-5] IP Address
10, 11	Value [6-7] Port Number
:	Value [8-11] IP Address (4 Bytes)
:	Value [12-13] Port Number (2 Bytes)
:	:

0x02BE Variant ID

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x02BE Variant ID
2, 3	Length 0x0002
4, 5	Value[0, 1] Variant ID 0x0000 ITU (Default)

0x02BF Version Number

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x02BF Version Number
2, 3	Length 0x0002
4, 5	Value[0, 1] Version Number 0x0002 H.323 V2 (Default)

0x02C0 Terminal Type

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x02C0 Terminal Type
2, 3	Length 0x0002
4	Value[0] 0x00 (Default) Terminal Type (MSB)
5	Value[1] Terminal Type (LSB) 0x3C GW with no MC (Default) 0x50** GW with MC but no MP 0x5A** GW with MC and Data MP 0x64** GW containing MC w/data & audio 0x6E** GW containing MC w/data, audio, & video **Not supported in this release.

0x02C1 Vendor ID

Used for Gateway H.323 ID registration as well as Vendor ID. The value is a ASCII null terminated string that spells out the default value "EXCEL-CSP" as indicated below.

Important! You cannot deconfigure the Vendor ID.

Used in:

0x0033 NPDI Universal ICB in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x2C1 Vendor ID
2, 3	Length 0x0A (Default)
4	Value[0] 0x45 E
5	Value[1] 0x58 X
6	Value[2] 0x43 C
7	Value[3] 0x45 E
8	Value[4] 0x4C L
9	Value[5] 0x2D --
10	Value[6] 0x43 C
11	Value[7] 0x53 S
12	Value[8] 0x50 P
13	Value[9] 0x00 (Null)

0x02C2 Gatekeeper Auto Discovery

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x2C2 Auto Discovery
2, 3	Length 0x0002
4, 5	Value[0, 1] Auto Discovery 0x0000 Disabled (Default) 0x0001 Enabled

0x02C3 Gatekeeper Auto Registration

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x2C3 Auto Registration
2, 3	Length 0x0002
4, 5	Value[0, 1] Auto Registration 0x0000 Disabled (Default) 0x0001 Enabled

0x02C4 Gateway E164

This TLV contains the alias address which is added to the outbound H.323 setup message if the *Outseize Control* message or *Route Control* message does not contain the calling party number.

You can deconfigure the Gateway ID by setting this value to Null.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x2C4 Gateway E164
2, 3	Length Variable (maximum 32-byte ASCII character strings)
:	Value [n] 0-9, #, * Last character must be 0. Default is Null

0x02C6 Gateway URL ID

Important! You can deconfigure the Gateway URL ID by setting it to Null.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x2C6 Gateway URL ID
2, 3	Length Variable (maximum 512-byte ASCII character strings)
:	Value [n] 0-9, #, * Last character must be 0. Default is Null

0x02C7 Gateway E-mail ID

Important! You can deconfigure the Gateway E-mail ID by setting it to Null.

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x2C7 Gateway E-mail ID
2, 3	Length Variable (maximim 512-byte ASCII character strings)
:	Value [n] 0-9, #, * Last character must be 0. Default is Null

0x02C8 H.225 T301 Timer

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x2C8 H225 T301 Timer configure
2, 3	Length 0x0002
4, 5	Value[0, 1] 0x4650 (Default) All timer values are used in units of 10 ms each.

0x02C9 H.225 T303 Timer

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x2C9 H225 T303 Timer Configure
2, 3	Length 0x0002
4, 5	Value[0, 1] 0x0190 (Default) All timer values are used in units of 10 ms each.

0x02CA H.225 T310 Timer

Used in:

VoIP Protocol Configure message*VoIP Protocol Query* message

Byte	Description
0, 1	Tag 0x2CA H225 T310 Timer Configure
2, 3	Length 0x0002
4, 5	Value[0, 1] 0x03E8 (Default) All timer values are used in units of 10 ms each.

0x02CB H.245 T101 Timer

Used in:

VoIP Protocol Configure message*VoIP Protocol Query* message

Byte	Description
0, 1	Tag 0x2CB H245 T101 Timer Configure
2, 3	Length 0x0002
4, 5	Value[0, 1] 0x0BB8 (Default) All timer values are used in units of 10 ms each.

0x02CC H.245 T103 Timer

Used in:

VoIP Protocol Configure message*VoIP Protocol Query* message

Byte	Description
0, 1	Tag 0x2CC H245 T103 Timer Configure
2, 3	Length 0x0002
4, 5	Value[0, 1] 0x0BB8 (Default) All timer values are used in units of 10 ms each.

0x02CD H.245 T105 Timer

Used in:

VoIP Protocol Configure message*VoIP Protocol Query* message

Byte	Description
0, 1	Tag 0x2CD H245 T105 Timer Configure
2, 3	Length 0x0002
4, 5	Value[0, 1] 0x01F4 (Default) All timer values are used in units of 10 ms each.

0x02CE H.245 T106 Timer

Used in:

VoIP Protocol Configure message*VoIP Protocol Query* message

Byte	Description
0, 1	Tag 0x2CE H245 T106 Timer Configure
2, 3	Length 0x0002
4, 5	Value[0, 1] 0x0BB8 (Default) All timer values are used in units of 10 ms each.

0x02CF H.245 T108 Timer

Used in:

VoIP Protocol Configure message*VoIP Protocol Query* message

Byte	Description
0, 1	Tag 0x2CF H245 T108 Timer Configure
2, 3	Length 0x0002
4, 5	Value[0, 1] 0x0BB8 (Default) All timer values are used in units of 10 ms each.

0x02D0 H.245 T109 Timer

Used in:

VoIP Protocol Configure message*VoIP Protocol Query* message

Byte	Description
0, 1	Tag 0x2D0 H245 T109 Timer Configure
2, 3	Length 0x0002
4, 5	Value[0, 1] 0x0BB8 (Default) All timer values are used in units of 10 ms each.

0x02D1 RAS GRQ Timer

Used in:

VoIP Protocol Configure message*VoIP Protocol Query* message

Byte	Description
0, 1	Tag 0x2D1 RAS GRQ Timer Configure
2, 3	Length 0x0002
4, 5	Value[0, 1] 0x0064 (Default) All timer values are used in units of 10 ms each.

0x02D2 RAS RRQ Timer

Used in:

VoIP Protocol Configure message*VoIP Protocol Query* message

Byte	Description
0, 1	Tag 0x2D2 RAS RRQ Timer Configure
2, 3	Length 0x0002
4, 5	Value[0, 1] 0x0064 (Default) All timer values are used in units of 10 ms each.

0x02D3 Gateway Technology Prefix

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

Byte	Description
0, 1	Tag 0x02D3 Gateway Technology Prefix
2, 3	Length Variable. 0x0003 (Default) Maximum 10 bytes.
:	Value [n] ASCII null terminated string. Default is as follows: 0x31 - 1 0x23 - # 0x00 - Null

0x02D4 H.323 Message Information Mask

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

The host uses this TLV to specify the additional H.323 message information in the *Request for Service with Data* message.

Byte	Description
0, 1	Tag 0x02D4
2, 3	Length 0x0004
4-7	Value[0] This field is a 32-bit mask. Each bit selects specific H.323 message fields and is listed from LSB to MSB. Bit 0 Reserved Bit 1 Reserved Bit 2 Reserved Bit 3 Reserved Bit 4 Selects Media Connection Address Bits 5-31 Reserved (Note: 0x10 = Select Media Connection Address) By default, all bits are set to 0. 0x00000000 - Null

0x02D5 H.245 Tunneling Configure

Used in:

VoIP Protocol Configure message

VoIP Protocol Query message

This TLV globally turn tunneling on and off. When this TLV is set to enable, the IP Signaling card accepts incoming H.323 calls with the H.245 tunneling feature.

Byte	Description
0, 1	Tag 0x02D5
2, 3	Length 0x0001
4-7	Value[0] 0 - Disabled (default) 1 - Enabled

0x05DC File Management Configuration

Use this TLV to specify the type of server to be used for recorded voice files. This TLV is used in the *Generic Card Configure* message and in the response to the *Generic Card Query* message for the DSP Series 2 card.

Byte	Description
0, 1	Tag 0x05DC File Management Configuration
2, 3	Length 0x0001
4	Value[0] 0x01 NFS

0x05DD Server Address

Use this TLV to specify the name and location of the NFS server used for voice files. This TLV is used in the *Generic Card Configure* message and in the response to the *Generic Card Query* message.

Byte	Description
0, 1	Tag 0x05DD Server Address
2, 3	Length Variable
4, 5	Value[0] Server ID (application-defined)
6-9	Value[1] Primary IP Address (all zeros deletes the socket)
10, 11	Value[2] Server Name Length
12-N	Value[3] Server Name
(2 bytes)	Value[4] Mount Name Length (max 32 bytes)
...	Value[5] Mount Name of Server

0x05DF Vocabulary Index File

Use this TLV to specific the server ID, path, and name for the Vocabulary Index File, which in turn tracks the name, ID, and location of recorded voice files.

This TLV is used in the *Generic Card Configure* message and in the response to the *Generic Card Query* message.

Byte	Description
0, 1	Tag 0x05DF Vocabulary Index File
2, 3	Length Variable
4, 5	Value[0, 1] Primary Server ID (ID of server the file is on)
6, 7	Value[2, 3] Secondary Server ID (For TFTP only--future). This is a secondary Server to look for the Vocabulary Index File (VIF) on. Any time the VIF is updated, the software tries the primary server, and if there is no response, attaches to the secondary server. When the VIF is updated, all caches are cleared.
8, 9	Value[4, 5] Primary File Name Length
10-N	Value[6-N] Primary File Name with Path, not including Mount name (max 100 bytes). The path must begin with one forward slash (/) or two backward slashes (\\)
(2 bytes)	Secondary Name Length (0=no secondary file)
(... -N)	Secondary File Name with Path, not including Mount name (max 100 bytes). The path must begin with one forward slash (/) or two backward slashes (\\)

0x05E0 File ID

Use this TLV to identify recorded voice files numerically.

This TLV is used in the *DSP Cache Modify*, *Play File Start*, and *Record File Start* messages.

Byte	Description
0, 1	Tag 0x05E0 File ID
2, 3	Length 0x0004
4-7	Value[0] File ID <u>RAN IDs 0x00100000 or higher:</u> These must be played alone (not chained), and the CSP must first be configured for NFS with the <i>Generic Card Configure</i> message. Files with these IDs are not stored on the card. <u>RAN IDs lower than 0x00100000:</u> You can chain these

0x05E1 File Format

Use this TLV to specify the file format and encoding format for recorded voice files for file IDs of 0x00100000 or higher in the *Play File Start* message and in the *Record File Start* message.

Also use this TLV in the *DSP Cache Modify* message when the Action field is set to Copy (the Encoding Format field is ignored; the encoding format of the file copy will always be the same format as the original file).

Byte	Description
0, 1	Tag: 0x05E1 File Format
2, 3	Length: 0x0002
4	Value[0]: File Format 0x00 - raw (packed raw binary data) 0x01 - vox 0x02 - wav
5	Value[1]: Encoding Format (for raw, vox and wav file types) 0x00 - G.711 A-law PCM 0x01 - G.711 μ -law PCM 0x02 - G.726 32 Kbps ADPCM 0x03 - 32 Kbps OKI ADPCM (Dialogic) 0x04 - 24 Kbps OKI ADPCM (Dialogic) 0x05 - 11025 Hz 16-bit 0x06 - 11025 Hz μ -law 0x07 - 11025 Hz A-law 0x08 - 11025 Hz 8-bit 0x09 - 8000 Hz 8-bit 0x0A - 8000 Hz linear 0x0B - G.726 40 Kbps 0x0C - G.726 24 Kbps 0x0D - G.726 16 Kbps (NOTE: The .wav format is derived from actual file)

0x05E2 File Location

Use this TLV to specify the location of recorded voice files.

This TLV is mandatory for file IDs of 0x00100000 or higher in the *Play File Start* message, and in the *Record File Start* message.

It is also mandatory in the *DSP Cache Modify* message when the Action field is set to Copy.

Byte	Description
0, 1	Tag 0x05E2 File Location
2, 3	Length Variable
4, 5	Value[0, 1] Primary Server ID
6, 7	Value[2, 3] Secondary Server ID
8-N	Value[4-N] Filename, not including Mount name (must begin with / or \\) 128 characters maximum

0x05E3 Gain

Use this TLV to modify the gain of recorded voice files.

This TLV is used in the following messages:

- *Play File Start*
- *Record File Start*
- *Play File Modify*
- *Record File Modify*

Byte	Description
0, 1	Tag 0x05E3 Gain
2, 3	Length 0x0001
4	Value[0] Gain, expressed in decibels Minimum -54 (0xCA) Maximum 24 (0x18) Default 0 (0x00)

0x05E4 Speed

Use this TLV to modify the speed of recorded voice files.

This TLV is used for all file IDs in the *Play File Start* message and in the *Play File Modify* message when the Action field is set to Modify.

Byte	Description
0, 1	Tag 0x05E4 Speed
2, 3	Length 0x0002
4, 5	Value[0, 1] Speed, expressed in percent Allowable values: 50 (%) (0x32) 75 (%) (0x4B) 100 (%) (0x64) Default Value 200 (%) (0xC8) 300 (%) (0x12C) 400 (%) (0x190)

0x05E5 Barge In

Use this TLV in the *Play File Start* and *Resource Create* message to enable barge-in for recorded files or a conference broadcast.

Byte	Description
0, 1	Tag: 0x05E5 Barge In
2, 3	Length: 0x0001
4	Value[0]: Barge In Flag Bit 0 0 = Barge In Disabled (default for Play files) 1 = Barge In Enabled (default for conferencing (broadcast)) Bit 1-7 Reserved (should be set to 0x00)

0x05E6 File Event Descriptor

Use this TLV in the *Play File Start Record File Start* messages to configure the generation of Call Processing Events related to the playing and recording of files..

Byte	Description
0, 1	Tag: 0x05E6 File Event Descriptor
2, 3	Length: 0x0001
4	Value[0]: File Event Descriptor Mask Default = 0x00 Bit 0 If set, Call Processing Event of "Play File Completed" (0x2C) is generated when the file finishes playing. Bit 1 If set, Call Processing Event of "Play File Started" (0x2B) is generated when the file begins playing. Bit 2 If set, Call Processing Event of "Record File Completed" (0x2E) is generated when the file finishes recording. Bit 3 If set, Call Processing Event of "Record File Started" (0x2D) is generated when the file begins recording. Bits 4-7 Reserved (must be 0x00)

0x05E7 Playback Repeat

Use this TLV in the *Play File Start* message to specify the number of times to repeat playing a recorded voice file or list of files, as well as the interval between repetitions and the total time for playing the file.

Byte	Description
0, 1	Tag 0x05E7 Playback Repeat
2, 3	Length 0x0006
4, 5	Value[0, 1] Number of times to repeat playing the file or list of files. 0xFFFF = Repeat Forever
6, 7	Value[2, 3] Interval between repetitions (100 ms steps) 0x05 (Default for 500 ms)
8, 9	Value[4, 5] Total time file is played (100 ms steps) 0x0000 = Disable timer (Default)

10,11	<p>Value[6,7] Length of File Played</p> <p>The length of the play time must be greater than or equal to the File time multiplied by the number of times the file is repeated.</p> <p>Also expressed as the formula below:</p> $Pt \geq (Ft + Dt) \times R$ <p>If the length of the file being played (or Ft) is 30 seconds, and the Interval or Delay time (Dt) is 10 seconds, and you want to repeat (R) the file 3 times, the Total time the file is played (Pt) must be ≥ 110 seconds.</p> <p>If this formula is followed, the Interval time (default of 500ms) can be configured to be more.</p> <p>However, if the total time (Pt) is LESS than the combined total of Ft and Dt, multiplied by the number of repetitions desired, the interval will not configure past the default of 500ms.</p>
-------	---

0x05E8 Beep Tone Parameters

Use this TLV in the *Record File Start* message to enable Beep Tone, as well as its frequency, duration, and amplitude.

Byte	Description
0, 1	Tag 0x05E8 Beep Tone Parameters
2, 3	Length 0x0005
4	Value[0] Beep Enable Flag 0 Disable 1 Enable (Default)
5, 6	Value[1, 2] Beep Frequency (Hz) Minimum 100 (0x0064) Maximum 3000 (0x0BB8) Default 1000 (0x03E8)
7	Value[3] Beep Duration (10 ms steps) Minimum 3 (0x03) Maximum 200 (0xC8) Default 20 (0x14)
8	Value[4] Beep Amplitude, in decibel milliwatts (dBm) Minimum -54 (0xCA) Maximum 3 (0x03) Default -15 (0xF1)

0x05E9 Initial Silence Timer

Use this TLV in the *Record File Start* message to set the Initial Silence Timer.

Recording does not begin until voice is detected. If the Initial Silence Timer expires before voice is detected, the recording attempt is aborted. For example, if the default setting of 2500 milliseconds is used, the recording attempt is aborted if no voice is detected in the first 2500 milliseconds.

Byte	Description
0, 1	Tag 0x05E9 Initial Silence Timer
2, 3	Length 0x0002
4, 5	Value[0-1] Timer (10 ms increments) Minimum 1 (0x0001) Maximum 65534 (0xFFFE) Disable timer 65535 (0xFFFF) Default 250 (0x00FA)

0x05EA Final Silence Timer

Use this TLV in the *Record File Start* message to specify how much silence it takes at the end of a voice recording to trigger the card to stop recording.

Byte	Description
0, 1	Tag 0x05EA Final Silence Timer
2, 3	Length 0x0002
4, 5	Value[0-1] Timer (10 ms increments) Minimum 1 (0x0001) Maximum 65534 (0xFFFE) Disable timer 65535 (0xFFFF) Default 250 (0x00FA)

0x05EB Maximum Record Timer

Use this TLV in the *Record File Start message* to set the maximum length of time allowed for recording a message.

Byte	Description
0, 1	Tag 0x05EB Maximum Record Timer
2, 3	Length 0x0002
4, 5	Value[0-1] Timer (10 ms increments) Minimum 1 (0x0001) Maximum 65534 (0xFFFE) Disable timer 65535 (0xFFFF) (Default)

0x05EC Skip Steps

Use this TLV in the *Play File Modify* message when the Action field is set to Skip Forward or Skip Backward.

Byte	Description
0, 1	Tag 0x05EC Skip Steps
2, 3	Length 0x0002
4, 5	Value[0-1] Steps (100 ms increments) Minimum 1 (0x0001) Maximum 65534 (0xFFFFE) Start of File 0 (0x0000) End of File 65535 (0xFFFFF)

0x05ED Dual Channel Record Option

Use this TLV when recording a conference with the *Record File Start* message to choose whether you want each channel to be saved to its own file or both channels to be saved to the same file.

Byte	Description
0, 1	Tag 0x05ED Dual Channel Record Option
2, 3	Length 0x0001
4	Value[0] 0x00 Sum Channels Together (Default) 0x01 Independent Channels (file location is the same, but filename is appended with "_2")

0x05EF Alarm Threshold Configure

Each DSP Series 2 main board can store up to three hours of files in a local cache. Each parameter has a threshold that generates an alarm. The M and N values provide a filter. M samples out of N must exceed the threshold for an alarm to occur. Files are stored in one-second blocks.

Used in:
 Generic Card Configure
 Generic Card Query
 VoIP Protocol Configure
VoIP Protocol Query

Byte	Description
0, 1	Tag 0x05EF Alarm Threshold Configure
2, 3	Length 0x0008
4	Value[0] Alarm Number 0 (0x00) Number of Reads 1 (0x01) Total Bytes Read 2 (0x02) Average Read Delay 3 (0x03) Maximum Read Delay 4 (0x04) Number of Writes 5 (0x05) Total Bytes Written 6 (0x06) Average Write Delay 7 (0x07) Maximum Write Delay 8 (0x08) Maximum Simultaneous Files Opened 9 (0x09) Average Simultaneous Files Opened 10 (0x0A) CPU Idle Time 11 (0x0B) VRA Process Delay 12 (0x0C) VRA IO Queue Delay
5	Value[1] Severity 0x00 Minor 0x01 Major
6	Value[2] M Value (Hits in window) Minimum 1 (0x01) Maximum 16 (0x10) Default 3 (0x03)
7	Value[3] N Value (Samples in window) (N >= M) Minimum 1 (0x01) Maximum 16 (0x10) Default 5 (0x05)
8-11	Value[4-7] Threshold Value (if this is a time value, it is in microseconds)

0x05F0 DSP 2 Main Board Memory Alarm Threshold Configure

Use this TLV to set the threshold in percentage of total blocks used, on a per-card basis. If recorded voice files are accessed through a Network File Server, then this value represents only the Temporary Cache memory space. Files are stored in one-second blocks.

If recorded announcements are accessed through a Network File Server, then this represents only the Temporary Cache memory space.

For the DSP Series 2 card, this TLV is used in the *Generic Card Configure* message and in the response to the *Generic Card Query* message.

Byte	Description
0, 1	Tag 0x05F0 DSP 2 Main Board Memory Alarm Threshold Configure
2, 3	Length 0x0006
4	Value[0] Parameter ID 0x00 Total Blocks
5	Value[1] Disable/Enable 0x00 Disabled (default) 0x01 Enable
6-9	Value[2-5] Threshold Value 1-100 (represents a percentage of the total available blocks)

0x05F1 DSP Temporary Memory Alarm Threshold Configure

Use this TLV in the *Generic Card Configure* message to set the threshold in percentage of total blocks, and to disable the alarm. This is set on a per-card basis, and the same threshold is used for all DSPs that are configured for File Playback/Record on that DSP Series 2 card. Files are stored in one-second blocks.

Byte	Description
0, 1	Tag: 0x05F1- DSP Temporary Memory Alarm Threshold Configure
2, 3	Length: 0x0006
4	Value[0]: Parameter ID 0x00 - Total Blocks
5	Value[1:]: Disable/Enable 0x00 - Disable (default) 0x01 - Enable
6-9	Value[2-5]: Threshold Value 1-100 (represents a percentage of the total available blocks)

0x05F2 Play File Queue

Use this TLV in the *Play File Start* message (0x011B) to add a file to the Play File Queue. The TLV supports all file IDs, and it works with both the Channel AIB and the Conference AIB.

If File A is playing and has not completed before you send File B, File B interrupts File A. This TLV causes File B to wait in queue so that File A can finish.

Byte	Description
0, 1	Tag: 0x05F2 - Play File Queue
2, 3	Length: 0x0002
4, 5	Value[0] 0x00 - Queue (Default) 0x01 - Override

0x05F3 Vocabulary Index Read Information

This TLV is sent in the response to the *Generic Card Query* message. It indicates the number of files in the Vocabulary Index File (VIF) and checks the integrity of the VIF through a checksum.

Byte	Description
0, 1	Tag: 0x05F3 - Vocabulary Index Read Information
2, 3	Length: 0x0008
4-7	Value[0-3] - Number of files in the Index file
8-11	Value[4-7] - Vocabulary Index File checksum

0x05F4 Statistics Update Timer

Use this TLV in the *Generic Card Configure* message to set how often system statistics are updated.

Byte	Description
0, 1	Tag: 0x05F4 - Statistics Update Timer
2, 3	Length: 0x0002
4, 5	Value[0, 1] - Timer Value Minimum: 1 (0x0001) Maximum: 168 (0x00A8) Default: 24 (0x0018)

0x05F5 Fixed Memory Statistics

This TLV is reported in the response to the *Statistics Query* message. It includes the total, average, and maximum blocks of fixed memory used on a specific DSP chip. A history of these statistics is not maintained.

Byte	Description
0, 1	Tag: 0x05F5 - Fixed Memory Statistics
2, 3	Length: 0x0008
4	Value[0] Module Number 0x00 - DSP Module 0 0x01 - DSP Module 1 0xFF - Mainboard
5	Value[1] Processor Number 0x00 - 0x03 0x00 for processor on Main board
6, 7	Value[2, 3] Total Blocks
8, 9	Value[4, 5] Average Blocks Used
10, 11	Value[6, 7] Max Blocks Used

0x05F6 DSP Series 2 Cache Statistics

This TLV is included in the response to the *Statistics Query* message. It reports cache accesses, hits, and misses, as well as average free blocks for a specific DSP chip. The history of these statistics is kept.

Byte	Description
0, 1	Tag: 0x05F6 Cache Statistics
2, 3	Length: 0x0010
4	Value[0] Module Number 0x00 - DSP Module 0 0x01 - DSP Module 1 0xFF - Main board
5	Value[1]: Processor Number Valid Range - 0-3 0x00 for main board processor
6-9	Value[2-3]: Cache Accesses
10-13	Value[6-9]: Cache Misses
14-17	Value[10-13]: Cache Hits
18, 19	Value[14, 15]: Average Free Blocks

0x05F7 DSP Function Statistics

This TLV is included in the response to the *Statistics Query* message. It reports average free channels, minimum free channels, and total requests for each function on a specific DSP chip on a DSP module. This data is repeated for the Txmt0, Rcv0, Txmt1, and Rcv1.

Byte	Description
0, 1	Tag:0x05F7 - DSP Function Statistics
2, 3	Length: 0x000C
4	Value[0]: Module Number 0x00 - DSP Module 0 0x01 - DSP Module 1
5	Value[1]: Processor Number Valid Range - 0x00 - 0x03
6	Value[2]: Sub-DSP Section 0x00 - Receive 0 0x01 - Transmit 0 0x02 - Receive 1 0x03 - Transmit 1
7	Value[3]: Function Number
8-11	Value[4-7]:Total Requests for Function
12, 13	Value[8, 9]: Average Free Channels
14, 15	Value[10, 11]: Minimum Free Channels

0x05F8 NFS Statistics

This TLV is included in the response to the *Statistics Query* message. It reports a wide range of NFS statistics for a specific DSP chip.

Byte	Description
0, 1	Tag: 0x05F8 - NFS Statistics
2, 3	Length: 0x005C
4	Value[0] Module Number 0x00 - DSP Module 0 0x01 - DSP Module 1 0xFF - Main board
5	Value[1]: Processor Number Valid Range - 0x00 - 0x03 0x00 for main board processor
6-9	Value[2-5] :Total Reads
10-13	Value[6-9]: Total Size of Reads
14-17	Value[10-13]: Minimum Read Time Delay
18-21	Value[14-17]: Maximum Read Time Delay
22-25	Value[18-21]: Average Read Time Delay
26-29	Value[22-25]: Total Writes
30-33	Value[26-29]: Total Size of Writes
34-37	Value[30-33]: Minimum Write Time Delay

38-41	Value[34-37]: Maximum Write Time Delay
42-45	Value[38-41]: Average Write Time Delay
46-49	Value[42-45]: Total Opens
50-53	Value[46-49]: Total Size of Opens
54-57	Value[50-53]: Minimum Open Time Delay
58-61	Value[54-57]: Maximum Open Time Delay
62-65	Value[58-61]: Average Open Time Delay
66, 67	Value[62, 63]: Simultaneous Files Being Played from Cache
68, 69	Value[64, 65]: Max Simultaneous Files Being Played from Cache
70, 71	Value[66, 67]: Average Simultaneous Files Being Played from Cache
72, 73	Value[68, 69]: Simultaneous Files Playing & Filling the 8260 Cache
74, 75	Value[70, 71]: Maximum Simultaneous Files Playing & Filling 8260 Cache
76, 77	Value[72, 73]: Average Simultaneous Files Playing & Filling 8260 Cache
78, 79	Value[74, 75]: Simultaneous Files Being Played Directly from NFS
80, 81	Value[76, 77]: Maximum Simultaneous Files Being Played Directly from NFS
82, 83	Value[78, 79]: Average Simultaneous Files Being Played Directly from NFS
84, 85	Value[80, 81]: Simultaneous Files Being Written to 8260 Cache
86, 87	Value[82, 83]: Maximum Simultaneous Files Being Written to 8260 Cache
88, 89	Value[84, 85]: Average Simultaneous Files Being Written to 8260 Cache
90, 91	Value[86, 87]: Simultaneous Files Being Written to the NFS Server
92, 93	Value[88, 89]: Maximum Simultaneous Files Being Written to the NFS Server
94, 95	Value[90, 91]: Average Simultaneous Files Being Written to the NFS Server

0x05F9 Query Statistics

Use this TLV in the *Statistics Query* message to indicate the Query Type and Sample Number.

Byte	Description
0, 1	Tag: 0x05F9 - Query Statistics
2, 3	Length: 0x0004
4, 5	Value[0, 1] Query Type 0x0400 - DSP Function Statistics 0x0401 - NFS Statistics 0x0402 - Fixed Memory Statistics 0x0403 - DSP Series 2 Cache Statistics 0x0404 - Resource Point Statistics 0x0405 - CPU Statistics 0x0406 - DSP 2 Overload Statistics 0xFFFF - Returns all valid Queries for the entity queried.
6, 7	Value[2-3]: Sample Number 0x0000 - Current Working Buffer 0x0001- 0x0010 - Saved copies [N-1], N-2] . . . 0xFFFE - Instantaneous values of used and free channels 0xFFFF - A copy that is cleared every time it is read

0x05FA Card Object

Use this TLV in the *Generic Card Configure* message to identify the card to be configured. The card uses this information to verify that the message is intended for it.

Byte	Description
0, 1	Tag: 0x05FA - Card Object
2, 3	Length: 0x0004
4, 5	Value[0, 1]: Card Type ID Refer to the Card Population Query 0x0007 message for a list of the card types.
6, 7	Value[2, 3]: Object ID 0x0000 - General Card Configuration 0x0001 - Conferencing Parameters 0x0003 - Alarm Configuration (Used in <i>Generic Card Query</i> message only) 0x0004 - DSP Series 2 Fax Parameters Query 0x0005 - Echo Cancel 0x0006 - PVD/AMD Parameters 0x0007 - Dial Pulse Detection Parameters

0x05FB Silence Threshold

Use this TLV in the *Record File Start* to set the Silence Threshold. When the Beep Amplitude is reached, silence is no longer assumed.

Byte	Description
0, 1	Tag: 0x05FB - Silence Threshold
2, 3	Length: 0x0001
4	Value[0] Beep Amplitude, in decibel milliwatts (dBm) Minimum: -54 (0xCA) Maximum: 3 (0x03) Default: -25 (0xE7)

0x05FC Temporary Record File Timer

Use this TLV in the *Record File Start* message to set the maximum amount of time that an internal recording will stay in the cache before expiring. The timer runs on a per file basis.

Byte	Description
0, 1	Tag: 0x05FC - Temporary Record File Timer
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Timer Value, in seconds Minimum: 0 (0x0000) Maximum: 65535 (0xFFFF) Default: 30 (0x001E)

0x05FD CPU Statistics

This TLV may be included in the response to the *Statistics Query* message.

Byte	Description
0, 1	Tag: 0x05FD CPU Statistics
2, 3	Length: 0x0034
4	Value[0]: DSP Module
5	Value[1]: DSP Chip
6	Value[2]: Percentage Idle
7	Value[3]: Percentage of time in the kernel task
8	Value[4]: Percentage of time in servicing interrupts
9	Value[5]: Percentage of time in the tnettask task
10	Value[6]: Percentage of time in the tdspx task
11	Value[7]: Percentage of time in the tdspr task
12	Value[8]: Percentage of time in the tnet task
13	Value[9]: Percentage of time in the trac task
14	Value[10]: Percentage of time in the tvrap task
15	Value[11]: Total percentage of time
16-19	Value[12-15]: Number of ticks idle
20-23	Value[16-19]: Number of ticks in the kernel task
24-27	Value[20-23]: Number of ticks in servicing interrupts
28-31	Value[24-27]: Number of ticks in the tnettask task
32-35	Value[28-31]: Number of ticks in the tdspx task
36-39	Value[32-35]: Number of ticks in the tdspr task
40-43	Value[36-39]: Number of ticks in the tnettask task
44-47	Value[40-43]: Number of ticks in the trac task
48-51	Value[44-47]: Number of ticks in the tvrap task
52-55	Value[48-51]: Total ticks

0x05FE Resource Point Statistics

This TLV may be returned in the response to the *Statistics Query* message. Note: TLV 0x05F9, query type 0404 is only valid while addressing the CSP Matrix Series 3 Card slot to query Resource Points. It does not work, if you address the DSP Series 2 card slot to query Resource Points.

Byte	Description
0, 1	Tag: 0x05FE Resource Point Statistics
2, 3	Length: 0x0010
4-7	Value[0-3]:Points Configured
8-11	Value[4-7]: Points Available
12-15	Value[8-11]: Minimum Points Available
16-19	Value[12-15]: Average Points Available

0x05FF DSP Cache Modify Action

Use this TLV in the *DSP Cache Modify* message.

Byte	Description
0, 1	Tag: 0x05FF DSP Cache Modify Action
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Action 0x0000 - Clear Specific Cache (also requires one or more File ID TLVs) 0x0001 - Clear All Caches 0x0002 - Copy Specific Cache (also requires the File ID, File Format, and File Location TLVs)

0x0600 NFS User Group ID

Use this TLV in the *Generic Card Configure* message if you want to modify the default NFS User Group ID information. You should send this TLV at the same time you are configuring the card for NFS (along with the File Management Configure TLV).

Byte	Description
0, 1	Tag: 0x0600 - NFS User Group ID
2, 3	Length: Variable
4, 5	Value[0, 1]: User ID (Default is 2001)
6, 7	Value[2, 3]: Group ID (Default is 100)
8, 9	Value[4, 5]: Local Name Length
10 . . .	Value[6 . . .]: Local Name

0x0601 NFS Poll Retries

Use this TLV in the *Generic Card Configure* message to set the number of polls the NFS server can miss before being marked out of service.

Byte	Description
0, 1	Tag: 0x0601 - NFS Poll Retries
2, 3	Length: 0x0004
4, 5	Value[0, 1]: NFS Poll Retries: Default = 0x0001
6, 7	Value[2, 3]: NFS Read Retries: Default = 0x0001

0x0602 Resource Type

Use this TLV to identify the resource type in the following messages:

Resource Create (0x0124)

Resource Modify (0x0125)

Resource Delete (0x0126)

Resource Connect (0x0127)

Byte	Description
0, 1	Tag: 0x0602 - Resource Type
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Resource Type 0x0100 - Conference 0x0105 - Send Fax 0x0106 - Receive Fax 0x0107 - Child Conference 0x0108 - Echo Cancel 0x0109 - Positive Voice Detection/ Answering Machine Detection (PVD/AMD) 0x010A - Send SMS 0x010B - Receive SMS 0x010C - Send FSK 0x010D - Receive FSK

0x0603 Channel/DSP Pool

Use this TLV in the *Resource Create* (0x0124) message to indicate the available DSP resources for a conference..

Byte	Description
0, 1	Tag: 0x0603 - Channel/DSP Pool
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Conference Location 0x0000 - Local DSP/Local Channel 0x0001 - Local DSP/Any Channel 0x0002 - Any DSP/Any Channel

0x0604 DTMF Clamping/Filtering Enable

Use this TLV to enable DTMF Clamping/Filtering for a file or a conference.

Play File Start (0x011B) (when starting a file)

Record File Start (0x011E) (when starting a recording)

Play File Modify (0x011C) (play file in progress)

Record File Modify (0x011F) (record file in progress)

Resource Create (0x0124) (when creating a conference)

Resource Modify (0x0125) (existing conference /or when modifying channel parameters connected to conference or child conference)

Resource Connect (0x0127) (when connecting channels to conference/moving channels from parent to child conference)

Byte	Description
0, 1	Tag: 0x0604 - DTMF Clamping/Filtering Enable
2, 3	Length: 0x0002
4, 5	Value[0, 1]: DTMF Clamping Mode 0x0000 - Disabled (Default) 0x0001 - Enabled

0x0605 EXS Conferencing Encoding Type

Use this TLV in the *Resource Create* (0x0124) message to specify the encoding type for a conference over the EXNET® ring.

Byte	Description
0, 1	Tag: 0x0605 - EXS Conferencing Encoding Type
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Encoding Type 0x0000 - u-law 0x0001 - A-law 0x0002 - Mixed

0x0606 Monitor Conference Enable

Use this TLV in the *Resource Create* (0x0124) message to create a Monitor Conference.

Byte	Description
0, 1	Tag: 0x0606 - Monitor Conference Enable
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Encoding Type 0x0000 - Disabled (Default) 0x0001 - Enable Monitor Conference

0x0607 Output Gain Control

Use this TLV to set the output gain, in decibels, for a conference.

It is used in the in the following messages

Resource Create (0x0124)

Resource Modify (0x0125)

Resource Connect (0x0127) .

Byte	Description
0, 1	Tag: 0x0607 - Output Gain Control
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Output Gain (1 db steps) Minimum: -40 (0xFFD8) Maximum: 0 (0x0000) Default: 0 (0x0000)

0x0608 Noise Gating Enable

Use this TLV to enable and disable noise gating in a conference.

It is used in the following messages:

Resource Create (0x0124)

Resource Modify (0x0125)

Resource Connect (0x0127)

Byte	Description
0, 1	Tag:0x0608 Noise Gate Enable
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Noise Gating 0x0000 - Disabled (Default) 0x0001 - Enabled

0x0609 Noise Gate Parameters

Use this TLV in the *Resource Create* (0x0124) message to set parameters for gating noise, including:

- the time constant over which the noise level is checked
- the allowable noise level
- noise gating sensitivity

If noise gating sensitivity is set too low for a conference, there is a risk of clipped speech; if it is set too high, there is a risk of noise bleeding through.

Byte	Description
0, 1	Tag: 0x0609 - Noise Gate Parameters
2, 3	Length: 0x0006
4, 5	Value[0, 1]: Time Constant (5 ms steps) Min: 10 (0x000A) Max: 100 (0x0064) Default: 35 (0x0023)
6, 7	Value[2, 3]: Maximum Noise Level (1 dBm steps) Min: -54 (0xFFCA) Max: -10 (0xFFF6) Default: -20 (0xFFEC)
8, 9	Value[4, 5]: Noise Gating Sensitivity 0x0001 - Low 0x0002 0x0003- Default 0x0004 0x0005 - High

0x060A Echo Suppression Enable

Use this TLV to enable and disable echo suppression in a conference.

It is used in the following messages

Resource Create (0x0124)

Resource Modify (0x0125)

Resource Connect (0x0127)

Byte	Description
0, 1	Tag: 0x060A Echo Suppression Enable
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Echo Suppression 0x0000 - Disabled (Default) 0x0001 - Enable

0x060B Echo Suppression Parameters

Use this TLV in the *Resource Create* (0x0124) message to set decibel parameters for suppressing echo in conferences. The higher quality the network, the lower you can set these parameters.

Byte	Description
0, 1	Tag: 0x060B Echo Suppression Parameters
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Echo Return Loss (1 db steps) Min: -54 (0xFFCA) Max: -6 (0xFFFA) Default: -10 (0xFF6)

0x060C Automatic Gain Control Enable

Use this TLV to enable and disable Automatic Gain Control.

It is used in the following messages

Resource Create (0x0124)

Resource Modify (0x0125)

Resource Connect (0x0127)

Byte	Description
0, 1	Tag: 0x060C Automatic Gain Control Enable
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Automatic Gain Control 0x0000 - Disable (Default) 0x0001 - Enable

0x060D Automatic Gain Control Input Level

Use this TLV in the *Resource Create* message (0x0124) to set the target input level in conferences.

Byte	Description
0, 1	Tag: 0x060D Automatic Gain Control Input Level
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Input Level (1 dBm steps) Min: -54 (0xFFCA) Max: -3 (0xFFFFD) Default: -22 (0xFFEA)

0x060E Automatic Gain Control Parameters

Use this TLV in the *Resource Create* message (0x0124) to set the time constant over which gain is measured in a conference.

Byte	Description
0, 1	Tag: 0x060E Automatic Gain Control Parameters
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Time Constant (5 ms steps) Min: 100 (0x0064) Max: 20000 (0x4E20) Default: 2000 (0x07D0)

0x060F Conference Failure Behavior

Use this TLV in the *Resource Connect* message (0x0127) to choose whether a channel is purged or parked when a conference fails.

Byte	Description
0, 1	Tag: 0x060F Conference Failure Behavior
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Conference Failure Behavior 0x0000 - Purge Channel (Default) 0x0001 - Park Channel

0x0610 Conference ID

This TLV is included in the response to the *Resource Create* message (0x0124) to identify a conference.

Byte	Description
0, 1	Tag: 0x0610 Conference ID
2, 3	Length: 0x0004
4-7	Value[0-3]: Conference ID Bits 0-9: Local Conference ID Bit 10: Conference Type 0 - Normal 1 - Monitor Bits (15,11): Node ID Bits (31,16): Reserved

0x0611 Forced Flag

Use this TLV in the *Resource Delete* message (0x0126) to choose whether a conference is deleted forcefully or gracefully.

Byte	Description
0, 1	Tag: 0x0611 Forced Flag
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Forced Flag 0x0000 - Graceful Delete 0x0001 - Forced Delete

0x0612 Connection Type

Use this TLV in the *Connect with Data* (0x0005) message to set up a one-way connection that is maintained during a matrix switchover. Do not use in the *Connect* (0x0000) message.

Use this TLV in the *Resource Connect* (0x0127) message to specify the connection type for a conferee.

Used in:

0x1E Generic PPL ICB in:

Connect with Data message

Resource Connect message

Byte	Description
0, 1	Tag: 0x0612 Connection Type
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Connection Type 0x0000 - 2-Way Connect (Default) 0x0001 - 1-Way Connect 0x0002 - 1-way Connect (Talk only)

0x0613 Conference Size

Use this TLV in the *Resource Create* (0x0124) message to specify the size of the conference.

Byte	Description
0, 1	Tag: 0x0613 Conference Size
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Conference Size 0x0002-0x0080

0x0614 Offset and Length

Use this TLV in the *Play File Start* message (0x011B) to begin playing a file at an offset from the beginning of the file, and to specify how long into the file to play so it doesn't have to play to the end.

Because the Offset and Length are in bytes, and because they represent bytes in a file, the application is responsible for managing the different encoding formats, and for how bytes should represent time.

You can use this TLV when queueing files, but you cannot use it when chaining files within a single message.

Byte	Description
0, 1	Tag: 0x0614 Offset and Length
2, 3	Length: 0x0008
4-7	Value[0-3]: Offset (bytes from the start of the file)
8-11	Value[4-7]: Length (bytes from the offset to play)

0x0615 Append or Replace

Use this TLV in the *Record File Start* message (0x011E) to specify whether to append to a file or to completely replace the file.

You can set the “File Not Found” behavior so that if you attempt to append to a file that is not found, you can create a new file or fail the open command.

The Append or Replace TLV is ignored for temporary records, which are always replaced. It is also ignored when copying a temporary record to a file.

Byte	Description
0, 1	Tag: 0x0615 Append or Replace
2, 3	Length: 0x0002
4	Append or Replace File 0 - Replace 1 - Append
5	“File Not Found” Behavior 0 - Fail the Open Command 1 - Create a New File

0x0616 DSP Series 2 Overload Statistics

This TLV is included in the response to the *Statistics Query* message (0x0121), in response to a DSP Overload Statistics query.

Byte	Description
0, 1	Tag: 0x0616 DSP Series 2 Overload Statistics
2, 3	Length: 0x001D
4	Value[0]: DSP Series 2 Module
5	Value[1]: DSP Chip
6-9	Value[2-5]: VRA Process Delay Process Time of Cached Play Files, in microseconds
10-13	Value[6-9]: Maximum VRA Process Delay Maximum Process Time of Cached Play Files, in microseconds.
14-17	Value[10-13]: Average VRA Process Delay Average Process Time of Cached Play Files, in microseconds.

18-21	Value[14-17]: VRA IO Queue Delay Process Time of Non-Cached Play and Record Files in microseconds.
22-25	Value[18-21]: Maximum VRA IO Queue Delay Maximum Process Time of Non-Cached Play and Record Files, in microseconds.
26-29	Value[22-25]: Average VRA IO Queue Delay Average Process Time of Non-Cached Play and Record Files, in microseconds.
30	Value[26]: Percent CPU Idle
31	Value[27]: Minimum Percent CPU Idle Time
32	Value[28]: Average Percent CPU Idle Time

0x0617 Alarm Threshold Query

Use this TLV in the *Generic Card Query* message (0x0123) to query the Alarm Threshold on the DSP Series 2 card.

Byte	Description
0, 1	Tag: 0x0617 Alarm Threshold Query
2, 3	Length: 0x0001
4	Value[0]: Alarm Number Being Queried 0x00 - 0x0F 0xFF - Request a Bit Mask of Enabled Alarms

0x0618 Alarm Threshold Enabled Report

Use this TLV in the *Generic Card Query* message (0x0123).

Byte	Description
0, 1	Tag: 0x0618 Alarm Threshold Enabled Report
2, 3	Length: 0x0008
4 - 7	Value[0-3]: A Bit Mask of Level 1 Alarms Enabled on this Card
8-11	Value[4-7]: A Bit Mask of Level 2 Alarms Enabled on this Card

0x0619 PVD Parameters

Use this TLV in the *Generic Card Configure* message to set card-level parameters for Positive Voice Detection.

Byte	Description
0, 1	Tag: 0x0619 - PVD Parameters
2, 3	Length: 0x0006
4, 5	Value[0, 1]: Time Constant Min. = 10 (0x000A) Max. = 100 (0x0064) Default = 35 (0x0023)
6, 7	Value[2, 3]: Maximum Noise Level (1db steps) Min. = -54 (0xFFCA) Max. = -10(0xFFFF6) Default = -20 (0xFFEC)
8, 9	Value[4, 5]: Noise Gating Sensitivity 0x0001 Low 0x0002 0x0003 Default 0x0004 0x0005 High

0x0620 AMD Parameters

Use this TLV in the *Generic Card Configure* message to set card-level parameters for Positive Voice Detection/Answering Machine Detection.

Byte	Description
0, 1	Tag: 0x0620 - AMD Parameters
2, 3	Length: 0x0016
4, 5	Value[0, 1]: Frequency Band 1 Minimum Detection Frequency (Hz) Min = 200 (0x00C8) Max = 3000 (0x0BB8) Default = 350 (0x015E) DISABLE = 0
6, 7	Value[2, 3]: Frequency Band 1 Maximum Detection Frequency 1 (Hz) Min = 200 (0x00C8) Max = 3000 (0x0BB8) Default = 350 (0x015E) DISABLE = 0
8, 9	Value[4, 5]: Frequency Band 2 Minimum Detection Frequency (Hz) Min = 200 (0x00C8) Max = 3000 (0x0BB8) Default = 750 (0x02EE) DISABLE = 0

10, 11	Value[6, 7]: Frequency Band 2 Maximum Detection Frequency (Hz) Min = 200 (0x00C8) Max = 3000 (0x0BB8) Default = 1600 (0x0640) DISABLE = 0
12, 13	Value[8, 9]: Frequency Band 3 Minimum Detection Frequency (Hz) Min = 200 (0x00C8) Max = 3000 (0x0BB8) Default = 1700 (0x06A4) DISABLE = 0
14, 15	Value[10, 11]: Frequency Band 3 Maximum Detection Frequency (Hz) Min = 200 (0x00C8) Max = 3000 (0x0BB8) Default = 3000 (0x0BB8) DISABLE = 0
16, 17	Value[12, 13]: Minimum Tone on time for declaration of tone present (10 ms) Min = 2 (0x0002) Max = 3000 (0x0BB8) Default = 15 (0x000F) DISABLE = 0
18, 19	Value[14, 15]: Minimum Tone on time for declaration of tone not present (10 ms) Min = 2 (0x0002) Max = 3000 (0x0BB8) Default = 15 (0x000F) DISABLE = 0
20, 21	Value[16, 17]: Minimum level of tone to declare present dBm Min = -54 (0xFFCA) Max = 3 (0x0003) Default = -20 (0xFFEC) DISABLE = 0
22, 23	Value[18, 19]: Minimum Voice on time for declaration of AMD_VOICE present (10 ms) Min = 2 (0x0002) Max = 3000 (0x0BB8) Default = 15 (0x000F) DISABLE = 0
24, 25	Value[20, 21]: Minimum Voice off time for declaration of AMD_VOICE Not present (10 ms) Min = 2 (0x0002) Max = 3000 (0x0BB8) Default = 15 (0x000F) DISABLE = 0

0x061B AMD Reports

Byte	Description
0, 1	Tag: 0x061B - AMD Reports
2, 3	Length: 0x0002
4, 5	Value[0, 1]: Bitmask Set a bit location to enable CPE reports. All enabled by default 0x000F. 0x0001 - Silence 0x0002 - PVD 0x0004 - Tones 0x0008 - AMD

0x061C Server State

This TLV is included in the response of the *Generic Card Query* (0x0123) message in response to a DSP Series 2 General Card Configuration.

Byte	Description
0, 1	Tag: 0x061C - Server State
2, 3	Length: 0x0003
4, 5 6	Value[0]: Server ID (Application Defined) Value[1]: Server State 0x01 - Server is configured 0x02 - Server is connected 0x03 - Server is trying to connect 0x04 - Server is disconnected 0x05 - Server is disconnected urgent

0x0640 Matrix/Host Sequence Number Size

Use this TLV in the *Generic Card Configure* (0x0122) message to set the Matrix/Host Sequence Number Size.

Byte	Description
0, 1	Tag: 0x0640 - Matrix/Host Sequence Number Size
2, 3	Length: 0x0001
4	Value[0]: Sequence Number Size 0x00 - 8-Bit (Default) 0x01 - 16-Bit

0x0641 Header Parameter Format

Use this TLV in the *Generic Card Configure* (0x0122) and *Resource Connect* (0x0127) messages to set the Header Parameter Format..

Byte	Description
0, 1	Tag: 0x0641 - Header Parameter Format
2, 3	Length: Variable
4	Value[0] Format 0x00 - Insert 0x01 - Overlay 0x02 - Replace (Default)
5	Value[1]: Format String Length
6-nn	Value[2-n]: Format String Default: " %D %T Remote ID: %R page %P of %M" Note: %D : insert date %T : insert time %L : insert local fax ID %R : insert remote fax ID %P : insert current page number %M : insert total page count (available for transmission header only)

Example:

If your string is "Excel%D%T" your output could be:

Excel 12/15/04, 3:42 PM

0x0642 T.30 Control Parameter Max Value

Use this TLV in the *Generic Card Configure* (0x0122) and *Resource Connect* (0x0127) messages to set the T.30 Control Parameter Max. Value.

Byte	Description
0, 1	Tag: 0x0642 T.30 Control Parameter Max Value
2, 3	Length: 0x0002
4, 5	Value[0,1]: Maximum rate permitted for fax operation. Limits advertised or selected transmission rates. 0x0000 - 2400 bps 0x0001 - 4800 bps 0x0002 - 7200 bps 0x0003 - 9600 bps 0x0004 - 12000 bps 0x0005 - 14400 bps (Default)

0x0643 T.30 Control Parameter Transmit Level

Use this TLV in the *Generic Card Configure* (0x0122) and *Resource Connect* (0x0127) messages to set the T.30 Control Parameter Transmit Level.

Byte	Description
0, 1	Tag: 0x0643 T.30 Control Parameter Transmit Level
2, 3	Length: 0x0002
4, 5	Value[0,1]: Transmission Level in tenths of dbm -400 to 0dbm. Min -400 (0xFE70) Max 0 Default -135 (0xFF79)

0x0644 T.30 Control Parameter ECM Enabled

Use this TLV in the *Generic Card Configure* (0x0122) and *Resource Connect* (0x0127) messages to enable T.30 Control Parameter ECM.

Byte	Description
0, 1	Tag: 0x0644 T.30 Control Parameter ECM Enable
2, 3	Length: 0x0002
4, 5	Value[0,1]: ECM 0x0000 - Disabled (Default) 0x0001 - Enabled

0x0645 T.30 Control Parameter Local Session ID

Use this TLV in the *Generic Card Configure* (0x0122) and *Resource Connect* (0x0127) messages to set the T.30 Control Parameter Local Session ID.

Byte	Description
0, 1	Tag: 0x0645 T.30 Control Parameter Local Session ID
2, 3	Length: Variable (max. 20)
4-n	Value[0-n]: D to be used as the local ID for the session " " (Default)

0x0647 Receive Modem Type

Use this TLV in the *Generic Card Configure* (0x0122) message to set the Receive Modem Type.

Byte	Description
0, 1	Tag: 0x0647 Receive Modem Type
2, 3	Length: 0x0002
4, 5	Value[0-1]: Modems to advertise in the DIS 0x0001 V27 0x0002 V29 0x0008 V17 0x000B (V.29, V.27, V.17) default

0x0648 Receive Resolution Type

Use this TLV in the *Generic Card Configure* (0x0122) message to set the Receive Resolution Type.

Byte	Description
0, 1	Tag: 0x0648 Receive Resolution Type
2, 3	Length: 0x0002
4, 5	Value[0-1]: Resolutions to advertise in the DIS (bitmask) 0x0004 RES_204x98 0x0008 RES_204x196 0x0010 RES_204x391 0x0020 RES_408x391 0x0040 RES_200x200 0x0080 RES_300x300 0x0100 RES_400x400 0x01FC (default)

0x0649 Receive Encoding Type

Used in: *Generic Card Configure* (0x0122) and *Generic Card Query* (0x0123) messages.

Byte	Description
0, 1	Tag: 0x0649 Receive Encoding Type
2, 3	Length: 0x0002
4, 5	Value[0,1]: Encodings to Advertise in the DIS (bitmask) 0x0001 MH (T.4, 1-D base mode) 0x0002 MR (T.4, 2-D encoded) 0x0003 MMR (T.6 encoding) (Default)

0x064B Receive Bad Line

Use this TLV in the *Generic Card Configure* (0x0122) and *Resource Connect* (0x0127) messages to set the action to take for the placement of bad lines in an image.

Byte	Description
0, 1	Tag: 0x064B Receive Bad Line
2, 3	Length: 0x0002
4, 5	Value[0,1]: Action to take for placement of bad lines in the image 0x0000 Do Nothing 0x0001 Repeat Last Good Line (Default) 0x0002 Blank Line with Tic Marks on Each Side 0x0003 Blank Line

0x064C Receive Page Size

Used in: *Generic Card Configure* (0x0122), *Generic Card Query* (0x0123), *Resource Connect* (0x0127) and *Resource Query* (0x012C) messages.

Byte	Description
0, 1	Tag: 0x064C Receive Page Size
2, 3	Length: 0x0002
4, 5	Value[0,1]: Page Sizes to Advertise 0x0001 - ISO A4 (210mm x 297mm) 0x0002 - JIS B4 (257mm x 364mm) 0x0004 - ISO A3 (297mm x 420mm) 0x0007 - (A4+B4+A3) (Default)

0x0651 Receive Enable ECM

Used in: *Generic Card Configure* (0x0122), *Generic Card Query* (0x0123), *Resource Connect* (0x0127), and *Resource Query* (0x012C) messages.

Byte	Description
0, 1	Tag: 0x0651 Receive Enable ECM
2, 3	Length: 0x0002
4, 5	Value[0,1]: Enable Error Correction Mode 0x0000 No (Default) 0x0001 Yes

0x0652 Receive Add Header

Used in: *Generic Card Configure* (0x0122), *Generic Card Query* (0x0123), *Resource Connect* (0x0127), and *Resource Query* (0x012C) messages. (Not supported in the current release.)

Byte	Description
0, 1	Tag: 0x0652 Receive Add Header
2, 3	Length: 0x0002
4, 5	Value[0,1]: Receive Add Header 0x0000 - Disable (Default) 0x0001 - Enable

0x0654 Receive Line Error Threshold

Used in: *Generic Card Configure* (0x0122) and *Generic Card Query* (0x0123) messages.

Byte	Description
0, 1	Tag: 0x0654 Receive line error threshold
2, 3	Length: 0x0002
4, 5	Value[0,1]: Percentage of line in error that forces an RTN 0-100 (Default = 10)

0x0655 Receive Timeout

Used in: *Generic Card Configure* (0x0122), *Generic Card Query* (0x0123), *Resource Connect* (0x0127), and *Resource Query* (0x012C) messages.

Byte	Description
0, 1	Tag: 0x0655 Receive Timeout
2, 3	Length: 0x0004
4-7	Value[0-3]: Length of T1, in milliseconds. Normally 35000. 35000 (Default)

0x0657 Receive Terminal ID

Used in: *Generic Card Configure* (0x0122), *Generic Card Query* (0x0123), *Resource Connect* (0x0127), and *Resource Query* (0x012C) messages.

Byte	Description
0, 1	Tag: 0x0657 Receive Terminal ID
2, 3	Length: Variable (Max 20)
4-nn	Value[0]: Terminal ID String (Default: “ ”)

0x065A Transmit Modem Type

Used in: *Generic Card Configure* (0x0122) and *Generic Card Query* (0x0123) messages.

Byte	Description
0, 1	Tag: 0x065A Transmit Modem Type
2, 3	Length: 0x0002
4, 5	Value[0,1]: Modems to Use (bitmask) 0x01 V27 0x02 V29 0x08 V17 0x0b (V27+V29+V17) (Default)

0x065B Transmit Resolution Type

Used in: *Generic Card Configure* (0x0122), *Generic Card Query* (0x0123), *Resource Connect* (0x0127), and *Resource Query* (0x012C) messages.

Byte	Description
0, 1	Tag: 0x065B Transmit Resolution Type
2, 3	Length: 0x0002
4, 5	Value[0,1]: Resolutions to advertise in the DIS (Bit Mask) 0x0001 - RES 75 x75 0x0002 - RES 150 x150 0x0004 - RES 204 x98 0x0008 - RES 204 x196 0x0010 - RES 204 x391(default) 0x0020 - RES 408 x391 0x0040 - RES 200 x200 0x0080 - RES 300 x300 0x0100 - RES 400 x400 0x0200 - RES 600 x600

0x065C Transmit Page Size

Used in: *Generic Card Configure* (0x0122), *Generic Card Query* (0x0123), *Resource Connect* (0x0127), and *Resource Query* (0x012C) messages.

Byte	Description
0, 1	Tag 0x065C Transmit Page Size
2, 3	Length 0x0002
4, 5	Value[0,1] Page Size to Support (Bit Mask) 0x0001 ISO A4 (210mm x 297mm) 0x0002 JIS B4 (257mm x 364mm) 0x0004 ISO A3 (297mm x 420mm) 0x0008 ISO A6 (864pels on 107mm or 1728pels on 107mm) 0x0010 ISO A5 (1216pels on 151mm) 0x0020 Letter 0x0040 Legal 0x0007 A4+B4+A3 (Default)

0x0661 Transmit Enable ECM

Used in: *Generic Card Configure* (0x0122), *Generic Card Query* (0x0123), *Resource Connect* (0x0127), and *Resource Query* (0x012C) messages.

Byte	Description
0, 1	Tag 0x0661 Transmit Enable ECM
2, 3	Length 0x0002
4, 5	Value[0,1] Enable Error Correction Mode 0x0000 No (Default) 0x0001 Yes

0x0662 Transmit Add Header

Used in: *Generic Card Configure* (0x0122), *Generic Card Query* (0x0123), *Resource Connect* (0x0127), and *Resource Query* (0x012C) messages. (Not supported in the current release.)

Byte	Description
0, 1	Tag 0x0662 Transmit Add Header
2, 3	Length 0x0002
4, 5	Value[0,1] Enable Transmit Header 0x0000 No 0x0001 Yes (Default)

0x0664 Transmit Enable CNG

Used in: *Generic Card Configure* (0x0122), *Generic Card Query* (0x0123), *Resource Connect* (0x0127), and *Resource Query* (0x012C) messages.

Byte	Description
0, 1	Tag 0x0664 Transmit Enable CNG
2, 3	Length 0x0002
4, 5	Value[0,1] Enable CNG Generation 0x0000 No 0x0001 Yes (Default)

0x0666 Transmit Timeout

Used in: *Generic Card Configure* (0x0122) and *Generic Card Query* (0x0123) messages.

Byte	Description
0, 1	Tag 0x0666 Transmit Timeout
2, 3	Length 0x0004
4-7	Value[0-3] Length of T1 in milliseconds Time in milliseconds (Default = 35000)

0x0668 Transmit dbm Level

Used in: *Generic Card Configure* (0x0122), *Generic Card Query* (0x0123), *Resource Connect* (0x0127), and *Resource Query* (0x012C) messages. (Not supported in the current release.)

Byte	Description
0, 1	Tag 0x0668 Transmit dbm Level
2, 3	Length 0x0002
4, 5	Value[0,1] dbm level for G.711 transmission in tenths of dbm. -400 to 0dbm. Min -400 (0xFE70) Max 0 Default -135 (0xFF79)

0x066A Transmit Terminal ID

Used in: *Generic Card Configure* (0x0122), *Generic Card Query* (0x0123), *Resource Connect* (0x0127), and *Resource Query* (0x012C) messages.

Byte	Description
0, 1	Tag 0x066A Transmit Terminal ID
2, 3	Length Variable (max 20)
4-nn	Value[0] Terminal ID String (Default: " ")

0x066C Fax Processing Events Set Register

Used in: *Generic Card Configure* (0x0122) and *Generic Card Query* (0x0123) messages.

Byte	Description
0, 1	Tag 0x066C Fax Processing Events Set Register
2, 3	Length 0x0002
4, 5	Value[0,1] Register Event Set (bitmask) (see details below) 0x0001 Phase A fax processing events 0x0002 Phase B fax processing events 0x0004 Phase C fax processing events 0x0008 Phase D fax processing events 0x0010 Phase E fax processing events 0x0020 Timer Events 0x0040 Data pump Events

0x066D Fax Configuration Query Type

Use this TLV in the *Generic Card Query* message with the Object Type of DSP 2 Fax Query.

Byte	Description
0, 1	Tag 0x066D Fax Configuration Query Type
2, 3	Length 0x0002
4, 5	Value[0, 1] Query Type 0x0001 - T.30 control parameters 0x0002 - Receive parameters 0x0004 - Transmit parameters

0x0673 Echo Cancel Tap Length

Use this TLV in the *Generic Card Configure* message to set card-level parameters for Echo Cancel.

Byte	Description
0, 1	Tag: 0x0673
2, 3	Length: 0x0002
4, 5	Value [0,1] Tap Length This is the length of the echo to be removed. Tail Length of the echo canceller FIR filter in number of samples. Allowed Options (8 samples = 1 millisecond) 0x0040 (64) = 8ms 0x0080 (128) = 16ms 0x0100 (256) = 32ms 0x0200 (512) = 64ms 0x0400 (1024) = 128ms

0x0674 Echo Cancel NLP Type

Use this TLV in the *Generic Card Configure* message to set card-level parameters for Echo Cancel.

Byte	Description
0, 1	Tag: 0x0674
2, 3	Length: 0x0001
4	Value [0] NLP Type This enables/disables the Non Linear Processor and, when the NLP is enabled, sets the type of Comfort Noise Generator used. 0x00 = Off 0x01 = On 0x02 = Random Noise CNG 0x03 = Hoth Noise CNG 0x04 = Maximum Suppression

0x0675 Echo Cancel ADAPT

Use this TLV in the *Generic Card Configure* message to set card-level parameters for Echo Cancel.

Byte	Description
0, 1	Tag: 0x0675
2, 3	Length: 0x0001
4	Value [0] This enables/disables the FIR filter coefficient adaptation. 0x00 = Disable 0x01 = Enable

0x0676 Echo Cancel Bypass

Use this TLV in the *Generic Card Configure* message to set card-level parameters for Echo Cancel.

Byte	Description
0, 1	Tag: 0x0676
2, 3	Length: 0x0001
4	Value [0] Bypass Enable 0x00 = Normal Operation; 0x01 = Bypass the echo canceller 0x02 = Bypass the echo canceller but still adapt its internal states and coefficients.

0x0677 Echo Cancel G.176 Modem Answer Detection

Use this TLV in the *Generic Card Configure* message to set card-level parameters for Echo Cancel.

Byte	Description
0, 1	Tag: 0x0677
2, 3	Length: 0x0001
4	Value [0] Enable/Disable 0x00 = Disabled 0x01 = Enable

0x0678 Echo Cancel NLP Threshold

Use this TLV in the *Generic Card Configure* message to set card-level parameters for Echo Cancel.

Byte	Description
0, 1	Tag: 0x0678
2, 3	Length: 0x0002
4	Value [0] Nonlinear processor threshold -32767 to +32767 (negative in 2's complement) Negative number indicates less aggressive NLP. Formula: $32767 * 10^{(-dB)/20}$ dB = value you enter

0x0679 Echo Cancel CNG Noise Threshold

Use this TLV in the *Generic Card Configure* message to set card-level parameters for Echo Cancel.

Byte	Description
0, 1	Tag: 0x0679
2, 3	Length: 0x0002
4	Value [0] CNG Noise Threshold Maximum near end noise level used to compute CNG level 0 to 32767 Formula: $32767 * 10^{(dBm-3)/20}$ dB = value you enter

0x067A Echo Cancel H Register Reset

Use this TLV in the *Resource Modify* message to reset the Echo Cancel Receiver.

Byte	Description
0, 1	Tag: 0x067A
2, 3	Length: 0x0001
4	Value [0] 0x00 = Disable 0x01 = Enable

0x067E Echo Cancel Comfort Noise Level

Use this TLV in the *Resource Connect* message to add comfort noise to RTP/ASR.

Byte	Description
0, 1	Tag: 0x067E
2, 3	Length: 0x0002
4, 5	Value [0, 1] 0x0000 - 0x0059 Default - 0x0032 This value determines the initial level of the comfort noise. The higher this value is, the lower the added noise is. Dialogic does not recommend values 0x0000 - 0x001E.

0x0687 - Enable RTP for Play/Record File

Use this TLV to enable RTP for Play/Record File in the following messages:

- *Play File Start*
- *Record File Start*

Byte	Description
0, 1	Tag: 0x0687 - Enable RTP for Play/Record File
2, 3	Length: 0x0001
4	Value [0] 0x00 - Disable 0x01 - Enable

0x0688 - RTP Record Mode

Use this TLV to set the RTP Record Mode in the following messages:

Play File Start

Record File Start

Byte	Description
0, 1	Tag: 0x0688 - RTP Record Mode
2, 3	Length: 0x0001
4	Value [0] 0x00 - Continuous Recording (default) Recording controlled with <i>Record File Start</i> and <i>Record File Stop</i> messages 0x01 - Recording with PVD RTP packets will only be sent during the time that Positive Voice is detected. Use in conjunction with: 0x0689 - Start/Stop Sending RTP Packets 0x068A - Calendar Time Offset for Sending RTP Packets

0x0689 - Start/Stop Sending RTP Packets

Use this TLV in the *Record File Modify* message to start or stop the sending of RTP packets with Positive Voice Detection.

To use this TLV, you must have previously sent a *Record File Start* message with the 0x0688 - RTP Record Mode TLV with the mode set to 0x01 - Recording with PVD.

Byte	Description
0, 1	Tag: 0x0689 - Start/Stop Sending RTP Packets
2, 3	Length: 0x0001
4	0x00 - Stop 0x01 - Start

0x068A - Calendar Time Offset for Sending RTP Packets

Use this TLV in the *Record File Modify* message to indicate the time at which positive voice is detected. The host will be notified with a Call Processing Event message.

To use this TLV, you must have previously sent a *Record File Start* message with the 0x0688 - RTP Record Mode TLV with the mode set to 0x01 - Recording with PVD.

This TLV is used with the Start/Stop Sending RTP Packets TLV (0x0689) in the *Record File Modify Message*.

Byte	Description
0, 1	Tag: 0x0689 - Calendar Time Offset for Sending RTP Packets
2, 3	Length: 0x0006
4-7	Calendar Time Offset (seconds since Jan. 1, 1970)
8, 9	Milliseconds

0x068B Input Gain Control

Use this TLV with the Channel AIB in the following messages to set the input gain, in decibels, for a conferee.

- *Resource Create* (0x0124)
- *Resource Modify* (0x0125)
- *Resource Connect* (0x0127)

Byte	Description
0, 1	Tag: 0x068B Input Gain Control
2, 3	Length: 0x0002
4-5	Input Gain (in 1 db increments) Min -40 (0xFFD8) Max +10 (0x000A) Default 0 (0x0000)

0x068C FAX Page Range

Use this TLV in the *Resource Connect* (0x0127) message to specify a page range for FAX transmission.

- To FAX a single page, enter same page number for Start Page and End Page.
- If the page range specified extends beyond the end of the TIFF file, the file will be transmitted to completion from the specified start page.

- If the specified start page is beyond the end of the TIFF file, then a Fax complete event shall be generated and no pages shall be transmitted.

Byte	Description
0, 1	Tag: 0x068C FAX Page Range
2, 3	Length: 0x0004
4, 5	Start Page Default = 0x01
6, 7	End Page Default = End of File

0x068D Transit Connection Mode

Use this TLV with the DSP Chip AIB in the following message to set the conference one-way (input only) temporarily, while the tone/file is being played.

- *Resource Create* (0x0124)
- *Resource Connect* (0x0127)

Byte	Description
0, 1	Tag: 0x068D Transit Connection Mode
2, 3	Length: 0x0002
4-5	Value [0,1] conference leg connection behavior while transmitting tone/file 0x0000: Remove leg (Default) 0x0001: Connect Input Only

0x0690 Frequency Shift Keying (FSK) Data

Use this TLV in the *Outseize Control* message.

Byte	Description
0, 1	Tag: 0x0690 Frequency Shift Keying (FSK) Data
2, 3	Length (variable): 0xnxxx
4, 5	Value[0, 1]: Data Type 0x0000 - ASCII Character 0x0001 - BCD Digit Pair 0x0002 - Raw Data
6, 7	Data Length (variable): 0xnxxx
8	Value[0]: Data 1 1st ASCII Character or 1st BCD Digit Pair
9	Value[0]: Data 2 2nd ASCII Character or 2nd BCD Digit Pair
nn	Value[0]: Data 1 nth ASCII Character or nth BCD Digit Pair

0x0691 Terminal Equipment Alerting Signal (TAS)

Use this TLV in the *Outseize Control* message.

Byte	Description
0, 1	Tag: 0x0691 Terminal Equipment Alerting Signal (TAS)
2, 3	Length (variable): 0x0002
4, 5	Value[0, 1]: Signal Type 0x0000 - DT-AS

0x0692 Subscriber Alerting Signal (SAS)

Use this TLV in the *Outseize Control* message.

Byte	Description
0, 1	Tag: 0x0692 Subscriber Alerting Signal (SAS)
2, 3	Length (variable): 0x0002
4, 5	Value[0, 1]: Signal Type 0x0000 - Call Waiting Tone

0x0693 FSK Data Transmission Type

Use this TLV in the *Outsize Control* message.

Byte	Description
0, 1	Tag: 0x0693 FSK Data Transmission Type
2, 3	Length (variable): 0x0002
4, 5	Value[0,1]: Data Transmission Type 0x0000 - FSK Transmission During Ringing 0x0001 - FSK Transmission Prior to Ringing 0x0000 - FSK Transmission without Ringing

0x0694 Short Message Service (SMS) Data

Use this TLV in the *Outsize Control* message.

Byte	Description
0, 1	Tag: 0x0694 Short Message Service (SMS) Data
2, 3	Length (variable): 0xnxxx
4	Value[0]: Message Type 0x01 - DLL_SMS_Data 0x02 - DLL_SMS_Error 0x03 - DLL_SMS_EST 0x04 - DLL_SMS_REL 0x05 - DLL_SMS_ACK 0x06 - DLL_SMS_NACK
5	Message Length (variable): 0xnn
6, nn	Value[0]: Payload

0x0695 FSK Modem Type

Use this TLV in the *Outsize Control* message.

Byte	Description
0, 1	Tag: 0x0695 FSK Modem Type
2, 3	Length (variable): 0x0002
4, 5	Value[0,1]: Data Transmission Type 0x0000 - ITU V.23 (default) 0x0001 - Bell 202 0x0002 - Bell 202 Chinese

0x0750 Dial Pulse Detection Parameters

Use this TLV in the *Generic Card Configure* and *Generic Card Query* messages.

Byte	Description
0, 1	Tag: 0x0750 - Dial Pulse Detection Parameters
2, 3	Length: 0x000A
4, 5	Value[0, 1]: Low Threshold Level - The sample level required to be below in order to declare sample as a Low Signal Default - 0x1388 (5000)
6, 7	Value[2, 3]: Low Threshold Count - The number of consecutive low samples to be considered a post Dial Pulse Low Default - 0x0011 (17)
8, 9	Value[4, 5]: Peak Threshold - The sample value required for a signal to be declared a Dial Pulse Peak Default - 0x36B0 (14000)
10, 11	Value[6, 7]: Maximum Transition Count - The maximum low to high transition time (samples). For a peak to be declared, the ramp time has to be less than this number of samples. Default - 0x7530 (30000)
12, 13	Value[8, 9]: Settle By Count: Number of consecutive samples that the signal must be below the low threshold to declare the signal as a dial pulse. Default - 0x0190 (400)

0x0751 - Play Continue On Digit Detection

The *Play File Start* (0x011B) supports the following new TLV.

Important! This TLV is not applicable for Play File message to conferences or child conferences.

Used in:

Play File Start (0x011B)

Byte	Description
0, 1	Tag 0x0751
2, 3	Length 0x0002
4	Value[0-3] 0 - Cancel on digit detection (default) 1 - Continue play on digit detection

0x09CA Add Global Title Group

This TLV is used to add a new Global Title (GT) group. It automatically activates the newly configured entries. After that, all GT entries that were pending to be added are brought into service. All GT entries that

were pending to be deleted are brought out of service and internal entry space becomes available for subsequent configurations. The index table is rebuilt automatically.

Used in:

SS7 SCCP TCAP Configure message

Byte	Description
0, 1	Tag 0x09CA Add Global Title Group
2, 3	Length 0x0009
4	Value[0] 0-127 Group ID
5	Value[1] Global Title Indicator 1-4 for ITU 1-2 for ANSI
6	Value [2] Translation Type 0-255 (0 = Default)
7	Value [3] Numbering Plan 0-15 (0 = Default)
8	Value [4] Nature of Address Indicator (NAI) 1-127 (0 = Default)
9	Value [5] Minimum Digits 1-24 The minimum number of digits that the GT entry of this group will contain. Must be greater than 0 and less than or equal to the Maximum Digits field.
10	Value [6] Maximum Digits 1-24 The maximum number of digits that the GT entry of this group will contain. Must be less than 24 and greater than or equal to the Minimum Digits field.

11	<p>Value [7] Group Attribute</p> <p>Bit 0 Maximum match 0 = Use minimum match rule for this group 1 = Use maximum match rule (also known as best match rule) for this group.</p> <p>Bit 1 UMTS CG (Not used in current release.) 0 = Calling Party address will not be changed when the Global Title translation results in "Route on GT." 1 = Calling Party address is changed when GT translation results in "Route on GT." The new calling party address is stored in SSN default parameter table. The UMTS CG bit will not take effect for connection oriented messages.</p> <p>Bit 2 ANSI TT4 0 = Not TT4 type 1 = TT4 type If this bit is set, no GT entry can be added into this group.</p> <p>Bit 3 Remove GT 0 = Do not remove GT after a successful translation. 1 = Remove GT after a successful translation.</p>
12	Reserved for future use.

0x09CB Delete Global Title Group

When you delete a group, all Global Title (GT) entries belonging to this group are removed. This TLV automatically activates the newly configured entries. After that, all GT entries that were pending to be added in are brought into service. All GT entries that were pending to be deleted are brought out of service and internal entry space becomes available for subsequent configurations. The index table is rebuilt automatically.

Used in:
SS7 SCCP TCAP Configure message

Byte	Description
0, 1	Tag 0x09CB Delete Global Title Group
2, 3	Length 0x0006

4	Value[0]	Group ID 1-127
5	Value [1]	Global Title Indicator 1-15
6	Value [2]	Translation Type 0-255 (0- Default)
7	Value [3]	Numbering Plan (NP) 0-15 (0 - Default)
8	Value [4]	Nature of Address Indicator (NAI) 1-127 (0 - Default)
9	Reserved for future use.	

0x09CC Add Global Title Entry

This TLV is used to add Global Title (GT) entries. It is not used for Global Title Translation (GTT) immediately. You can configure multiple GT entries before you activate them using the 0x09CE Build Index Table.

These newly configured GT entries are stored on the SS7 card but do not take effect until the SS7 card receives the Build Index command.

You can add 1,000 GT entries at a time up to a maximum of 100,000 for an SS7 card. If you configure GT entries after exceeding this maximum, you will receive the error code 0x552D. When the GT entry has been in the Active GT table you will receive the error code 0x5529. When the GT entry has been Standby GT table you will receive the error code 0x5528.

Used in:

SS7 SCCP TCAP Configure message

Byte	Description	
0, 1	Tag	0x09CC Add Global Title Entry
2, 3	Length	Variable
4	Value[0]	Group ID 1-127

5	Value [1]	Entry Attribute Bit 0 - Wild Card bit 0 Non-Wild Card GT 1 Wild Card GT Bits 1-7 Not Used.
6		Reserved for future use.
7	Value [2]	Global Title Address Information Length 1-24 digits
:	Value [n]	Global Title Address Information (GTAI) See Table below for format.
:	Value [n]	Translation Result Option 00 Single translation result 01 Double translation results, working in Active Standby Mode 02 Double translation results, working in Load Sharing Mode
:	Value [n]	Translation Result 1 See Table below.

Table 4-1 Global Title Address Information Format

Byte	Bits: 7 6 5 4	Bits: 3 2 1 0
0	Second digit	First Digit
1	Fourth Digit	Third Digit
:		
N	(M+1)th Digit	Mth Digit

Table 4-2 Translation Result Format

Byte	Field Name	Description
N	Translation Result Length	
N+1	Translation Result Format	Four formats: 00: RI + PC 01: RI+PC+SSN 02: RI+PC+GT 03: RI+PC+SSN+GT
N+2	Routing Indicator	00: Route on GT 01: Route on SSN
N+3 - N+6	Point Code	
If (N+1)Byte=0x01 RI+PC+SSN		
N+7	Subsystem Number	
If (N+1) Byte = 0x02 RI+PC+GT		
N+7	Global Title	See Table 4-3 below.
If (N+1) Byte=0x03 RI+PC+SSN+GT		
N+7	Subsystem Number	
N+8 - ...	Global Title	See Table 4-3 below.

Table 4-3 Global Title Format

Byte	Field	Description
N	GTI	Global Title Indicator Range: 1-15
N+1	GT Content Length	Range: 1-24
N+2	GT Content	Global Title Content

0x09CD Delete Global Title Entry

After the deleted Global Title (GT) Entry TLV is executed, the deleted GT Entry is not out of service immediately. It is used to do GTT until the index table is built which remove these GT entries from the GT table.

Used in:

SS7 SCCP TCAP Configure message

Byte	Description
0, 1	Tag 0x09CD Delete Global Title Entry
2, 3	Length Variable
4	Value[0] Group ID 1-127 Already configured.
5	Value [1] Entry Attribute (ATT) Bit 0 - Wild Card bit 0 = Non-Wild Card GT 1 = Wild Card GT Bits 1-7 Not Used.
6	Reserved for future use.
7	Value [2] Global Title Address Information Length 1-24 digits
:	Value [n] Global Title Address Information (GTAI) See Table 4-1 for format.

0x09CE Build Index Table

This TLV is used to activate configured GT entries. After the index table is built, all new added GT entries are in service and all deleted GT entries are out-of-service.

This TLV automatically activates the newly configured entries. After that, all GT entries that were pending to be added are brought into service. All GT entries that were pending to be deleted are brought out of service and internal entry space becomes available for subsequent configuration.

Used in:
SS7 SCCP TCAP Configure message

Byte	Description
0, 1	Tag 0x09CE Build Index Table
2, 3	Length 0x0001
4	Value[0] 1 - Build Index All other values reserved for future use.

0x09CF Global Title Group Query

Use this query type to query a Global Title (GT) Group.

Used in:
SS7 SCCP/TCAP Query message

Byte	Description
0, 1	Tag 0x09CF Global Title Group Query
2, 3	Length 0x0001
4	Value[0] Group ID 0-127

0x09D0 Global Title Entry Query

Use this query type to query a Global Title (GT) Entry.

Used in:
SS7 SCCP/TCAP Query message

Byte	Description
0, 1	Tag 0x09D0 Global Title Entry Query

2, 3	Length	0x0001
4	Value[0]	Group ID 0-127
5	Value[1]	GT Attribute Bit 0 - Wildcard bit 0 Non-wildcard GT 1 Wildcard GT Bits 1-7 Not Used.
6	Reserved for future use.	
7	Value[2]	Global Title Address Information Length 1-24 digits
:	Value[n]	Global Title Address Information (GTAI) See Table 4-1.

0x09D1 Global Title Group Query Response

Used in:

SS7 SCCP/TCAP Query message

Byte	Description	
0, 1	Tag	0x09D1 Global Title Group Query Response
2, 3	Length	0x0009 Note: If you attempt to query either a Global Title Group that does not exist, the TLV length in the response will be one byte consisting of null data.
4	Value[0]	State 0 Group Idle 1 Group In Service
5	Value[1]	Stack ID -State this group belongs to.
6	Value[2]	GTI GT Indicator

7	Value[3]	TT	Translation Type
8	Value[4]	NP	Numbering Plan
9	Value[5]	NAI	Nature of Address Indicator
10	Value[6]	Minimum Digits - The minimum number of digits the GT Entry of this group contains	
11	Value[7]	Maximum Digits - The maximum number of digits the GT Entry of this group contains	
12	Value[8]	<p>Group Attribute</p> <p>Bit 0 - Maximum Match 0 Use minimum match rule for this group 1 Use the maximum match rule for this group (best match rule).</p> <p>Bit 1 - UMTS CG (Not used in current release.) 0 Calling Party address will not be changed when GT translation results are "Route on GT." 1 Calling Party address is changed when GT translation results are "Route on GT." The new calling party address is stored in SSN default parameter table.</p> <p>Bit 2 - ANSI TT4 0 Not TT4 Type 1 TT4 Type If this bit is set, no GT entry can be added to this group.</p> <p>Bit 3 - Remove GT 0 Do not remove GT after a successful translation. 1 Remove GT after a successful translation.</p>	

0x09D2 Global Title Entry Query Response

Used in:

SS7 SCCP/TCAP Query message

Byte	Description
0, 1	Tag 0x09D2 Global Title Entry Query Response
2, 3	Length Variable Note: If you attempt to query either a Global Title Entry that does not exist, the TLV length in the response will be one byte consisting of null data.
4	Value[0] State 0 = Idle 1 = In service 2 = Pending_Add 3 = Pending_Del
5	Value[1] Translation Result Option
6	Value[2] Translation Result 1 See Table Translation Result Format
7	Value[3] Translation Result 2 (If it exists.) See Table Translation Result Format .

0x2710 Universal Protocol Data, Single

Use this TLV to specify the universal protocol data.

Used in:

PPL Table Download message

Byte	Description
0, 1	Tag 0x2710
2	Value[0] Data Format (See Note 1)
3	Value[1] Offset (See Note 3)
4-7	Value[2-5] Reserved
:	Value[:] Universal Protocol Data (See Note 2)
:	Value[:] Mask (See Note 2)

0x2710 Universal Protocol Data, Range

Use this TLV to specify the universal protocol data.

Used in:

PPL Table Download message

Byte	Description
0, 1	Tag 0x2710
2	Value[0] Data Format (See Note 1)
3	Value[1] Offset (See Note 3)
4-7	Value[2-5] Reserved
:	Value[:] To Universal Protocol Data (See Note 2)
:	Value[:] From Universal Protocol Data (See Note 2)
:	Value[:] Mask (See Note 2)

0x2712 Congestion Level (Interworking Only)

Used in:

NPDI Universal ICB (0x0033)

Byte	Description
0, 1	Tag 0x2712
2, 3	Length 0x0002
4	Value[0] Congestion Level 0x0000 Receiver ready 0x0001 Congestion level 1 exceeded 0x0002 Congestion level 2 exceeded 0x000F Receiver not ready

0x2716 Call Identity (Interworking Only)

Used in:

NPDI Universal ICB (0x0033)

Byte	Description
0, 1	Tag 0x2716
2, 3	Length Variable
4-6	Value[0] SS7 Point Code
7,8	Value[1] Number of Call Identity Bytes
9	Value[2] Call Identity

0x2717 Called Party Number

Use this TLV to specify the called party number. This TLV and be used in Interworking and Standard environments.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Route Control message

Byte	Description
0, 1	Tag 0x2717
2, 3	Length Variable
4	Value[0] Type of Number
5	Value[1] Numbering Plan Identifier and Reverse Charge Indicator
6	Value[2] Number of Digits (N)
7	Value[3] BCD Digit Pair 1
:	Value[4] BCD Digit Pair 2
:	Value[(N+1)/2] Half as many BCD pairs as digits.

0x2718 Calling Party Number (Connected Number)

Use this TLV to specify the calling party number. This TLV can be used in both Interworking and Standard environments.

The Screening and Presentation Indication byte and Numbering Plan Identifier byte are SS7 Called Party Number and Calling Party Number parameters defined in ANSI T1.113 and ITU Q.763.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Route Control message

Byte	Description
0, 1	Tag 0x2718
2, 3	Length Variable
4	Value[0] Type of Number
5	Value[1] Screening and Presentation Indication

6	Numbering Plan Identifier
7	Calling Party Category
8	Number of Digits (N)
9	BCD Digit Pair 1
:	BCD Digit Pair 2
:	:

**0x271A Circuit State
(Interworking Only)**

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x271A
2, 3	Length Variable
4	Value[0] Coding Standard 0x00 CCITT Standardized Coding 0x01 ISO/IEC Standard 0x02 National Standard 0x03 Standard specific to identified location
5, 6	Value[1] Circuit State 0x0000 IDLE 0x0001 Locally blocked 0x0002 Remotely blocked 0x0003 Locally and remotely blocked 0x0004 Circuit incoming busy 0x0005 Circuit outgoing busy 0x0006 Locally blocked by hardware 0x0007 Remotely blocked by hardware 0x0008 Locally and remotely blocked by hardware All other values are reserved.

**0x271B Cause Indicators
(Interworking Only)**

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x271B
2, 3	Length Variable
4	Value[0] Coding Standard

5	Value[1] Location
6	Value[2] Cause Value
7,8	Value[3] Length of Diagnostics
9	Value[4] Diagnostics Data 1
10	Value[5] Diagnostics Data 2
11	Value[6] Diagnostics Data [...]

0x2725 Forward Call Indicators (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2725
2, 3	Length Variable
4-8	Value[0] First byte indicator Bit 1 - Incoming International Call Indicator 0 Not an incoming international call 1 Incoming international call Bit 2 - End-to-End Method Indicator 00 No end-to-end method available 01 Pass along method available 10 SCCP Method available 11 Pass along and SCCP method available Bit 3 - Interworking Indicator 0 No interworking encountered (SS7 all the way) 1 Interworking encountered Bit 4 - IAM Segmentation Indicator 0 No indication 1 Additional information sent in an unsolicited information message Bit 5 - ISUP Indicator 0 ISUP not used all the way 1 ISUP used all the way Bit 6 - ISUP Preference Indicator 00 ISUP Preferred all the way 01 ISUP not required all the way 10 ISUP required all the way 11 Spare
5	Value[1] Second byte indicator Bit 1 - ISDN Access Indicator 0 Originating access non-ISDN 1 Originating access ISDN Bits 2-3 - SCCP Method Indicator 00 No indication 01 Connectionless method available 10 Connection oriented method available 11 Connectionless and connection oriented method available Bit 4 - Spare Bits 5-8 - Reserved for National Use

0x2727 Generic Digit Information (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2727
2, 3	Length Variable
4	Value[0] Acknowledgement Indicator 0x00 No ending acknowledgement required. 0x01 Ending acknowledgement required. 0xF0 to 0xFD User Defined 0xFF Not set or Unknown All the other values are reserved.
:	:
:	Value[1] Number of Digits
:	Value[2] Digit 12 in BCD pair
:	Value[3] Digit 34 in BCD pair

0x2729 Generic Number (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2729
2, 3	Length Variable
4	Value[0] Number qualifier
5	Value[1] Type of number
6	Value[2] Screening and presentation indication
7	Value[3] Number plan identifier
8	Value[4] Calling Party Category)
9	Value[5] Number of Digits
10	Value[6] BCD Digit Pair 1
11	Value [7]BCD Digit Pair 2
:	:

0x272A Generic Reference Number (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x272A
2, 3	Length Variable
4	Value[0] Reference MSB, LSB
:	:
:	Value[n] Reference MSB, LSB

0x272B Hop Counter (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x272B
2, 3	Length Variable
4	Value[0] Hop Counter Bits 1-5 Hop Counter Bits 6-8 Spare

0x2737 Original Called Number (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2737
2, 3	Length Variable
4	Value[0] Type of Number
6	Value[1] Screening and Presentation Indicators
7	Value[2] Numbering plan indicator
8	Value[3] Calling Party Category
9	Value[4] Number of digits
10	Value[5] BCD digit pair 1
11	Value[6] BCD digit pair 2
:	:

0x273C Redirecting Number (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x273C
2, 3	Length Variable
4	Value[0] Type of Number
5	Value[1] Redirection Reason
6	Value[2] Screening and Presentation Indicators
7	Value[3] Reverse charge and numbering plan indicator
8	Value[4] Category
9	Value[5] Number of digits
10	Value[5] BCD digit pair 1
11	Value[6] BCD digit pair 2
:	:

0x273D Redirection Information (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x273D
2, 3	Length Variable
4	Value[0] Type of Number
5	Value[1] Redirecting indicator / original redirection reason
6	Value[2] Redirection counter / redirection reason

0x273E Redirection Number (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x273E
2, 3	Length Variable
4	Value[0] Type of Number
5	Value[1] Redirection indicator / original redirection reason
6	Value[2] Redirection counter / redirection reason

0x2745 Transit Network Selection (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2745
2, 3	Length Variable
4	Value[0] Type of network 0x00 CCITT standardized identification 0x02 National network 0x03 International network 0x04 User specified The rest of the values are reserved.
	Value[1] Network identification plan 0x00 Unknown 0x01 Carrier identification code 0x03 Data network identification code 0x06 Land mobile network identification code The rest of the values are reserved.

0x2749 Bearer Capability (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2749
2, 3	Length Variable
4	Value[0] First byte of Q.931 Bearer Capability

0x274E NPDI Message Type

Use this TLV in the NPDI Universal ICB (0x0033) to indicate the message type. Most protocols require that you use the Message Type as the first TLV in the ICB. This TLV can be used in Interworking and Standard environments.

Used in:

NPDI Universal ICB (0x0033)

*Request for Service with Data message, Route Control message
Outseize Control message, Channel Released with Data message*

Byte	Description
0, 1	Tag: Message Type TLV (0x274E)
2, 3	Length: 0x0002
4, 5	Value [0.2]: Message Type 0x0001 Call Proceeding 0x0002 Connect 0x0003 Connect Acknowledge 0x0004 Progress 0x0005 Setup 0x0006 Setup Acknowledge 0x0007 Resume 0x0008 Resume Acknowledge 0x0009 Resume Reject 0x000A Suspend 0x000B Suspend Acknowledge 0x000C Suspend Reject 0x000D Alerting 0x000E Subsequent Address Message 0x000F Release 0x0010 Release Complete 0x0011 Information Request 0x0012 Information Response 0x0013 Identification Request 0x0014 Identification Response 0x0015 Forward Transfer 0x0016 Exit Message 0x0017 Status 0x0018 Status Enquiry 0x0019 Establish 0x002A Establish Acknowledgment 0x002B Signal 0x002C Signal Acknowledgment 0x002D Disconnect Complete 0x002E Port Control 0x002F Port Control Acknowledgment 0x0030 Common control 0x0031 Common control Acknowledgment 0x0032 Resource Unavailable 0x0033 User Information 0x0034 Segment 0x0035 Congestion Control 0x0036 Information 0x0037 Notify

0x274F Progress Indicator (Interworking Only)

Used in:

NPDI Universal ICB (0x0033)

Byte	Description
0, 1	Tag 0x274F
2, 3	Length Variable
4	Value[0] Coding Standard 0x00 CCITT Standardized Coding 0x01 ISO/IEC Standard 0x02 National Standard 0x03 Standard specific to identified location
5	Value[1] Location 0x00 User 0x01 Private network serving the local user 0x02 Public network serving the local user 0x04 Public network serving the remote user 0x05 Private network serving the remote user 0x06 Network beyond the Interworking point

6	<p>Value[2] Progress Descriptor</p> <p>0x00 (or 0x80*) Call is not end to end of same protocol 0x02 (or 0x82*) Destination address is non-ISDN 0x03 (or 0x83*) Origination address is non-ISDN 0x04 (or 0x84*) Call has returned to the ISDN 0x05 (or 0x85*) Interworking has occurred and resulted in telecomm service change 0x08 (or 0x88*) In-band information or an appropriate pattern is now available 0x40 (or 0xC0*) SCCP method available 0x60 (or 0xE0*) Pass along method available 0x20 (or 0x60*) Pass along and SCCP available 0xxxxxx Incoming half echo control device not included 1xxxxxx Incoming half echo control device included</p> <p>* when Echo device is included</p>
7	<p>Value[3] Charge Indicator</p> <p>0x00 No indication 0x01 No charge 0x02 Charge</p>
8	<p>Value[4] Called Party's Status Indicator</p> <p>0x00 No indication 0x01 Subscriber free 0x02 Connect when free</p>
9	<p>Value[5] Holding Indicator</p> <p>0x00 Holding not requested 0x01 Holding requested</p>
10	<p>Value[6] SCCP Method Indicator</p> <p>0x00 No indication 0x01 Connectionless method available 0x02 Connection oriented method available 0x03 Both connectionless and connection oriented method available</p>
:	<p>Optional Bytes</p> <p>For future enhancements in protocol. Also can be used to customize the protocol with other information with this packet.</p>

**0x2750 Display
(Interworking Only)**

Used in:
0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2750
2, 3	Length Variable
4:	Value[1] Display Information (IA5/ASCII NULL terminated characters)

**0x2751 High Layer Compatibility
(Interworking Only)**

Used in:
0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2751
2, 3	Value[0] Length
4, 5	Value[1] Coding Standard 0x00 CCITT Standardized Coding 0x01 ISO/IEC Standard 0x02 National Standard 0x03 Standard specific to identified location
6	Value[2] Interpretation 0x04 First (primary or only) high layer characteristic identification to be used in the call. All other values are reserved.
7	Value[3] Presentation Method of the Protocol Profile 0x01 High layer protocol profile. All other values are reserved.

8	Value[4] High layer characteristics identification
0x01	Telephony
0x04	Facsimile Group 2/3 (Recommendation F.182)
0x21	Facsimile Group 4 Class I (Recommendation F.184)
0x24	Teletex service, basic and mixed mode of operation (Recommendation F.230 and facsimile service group 4, Classes II and III (Recommendation F.184)
0x28	Teletex service, basic and processable mode of operation (Recommendation F.220 [71]).
0x31	Teletex service, basic mode of operation. (Recommendation F.200 [72]).
0x32	Syntax based Videotex. (Recommendations F.300 [73] and T.102 [74]).
0x33	International Videotex Interworking via gateways or Interworking units (Recommendations F.300 and T.101 [75]).
0x35	Telex service (Recommendation F.60 [76]).
0x38	Message Handling Systems (MHS) (X.400 Series Recommendations [77]).
0x41	OSI Application (X.200 Series Recommendations [78]).
0x60	Audio Visual (Recommendation F.721 [79])
	All the other values are reserved

9	<p>Value[5] Extended high layer characteristics identification</p> <p>0x01 Telephony</p> <p>0x04 Facsimile Group 2/3 (Recommendation F.182)</p> <p>0x21 Facsimile Group 4 Class I (Recommendation F.184)</p> <p>0x24 Teletex service, basic and mixed mode of operation (Recommendation F.230 and facsimile service group 4, Classes II and III (Recommendation F.184)</p> <p>0x28 Teletex service, basic and processable mode of operation (Recommendation F.220 [71]).</p> <p>0x31 Teletex service, basic mode of operation. (Recommendation F.200 [72]).</p> <p>0x32 Syntax based Videotex. (Recommendations F.300 [73] and T.102 [74]).</p> <p>0x33 International Videotex Interworking via gateways or Interworking units (Recommendations F.300 and T.101 [75]).</p> <p>0x35 Telex service (Recommendation F.60 [76]).</p> <p>0x38 Message Handling Systems (MHS) (X.400 Series Recommendations [77]).</p> <p>0x41 OSI Application (X.200 Series Recommendations [78]).</p> <p>0x60 Audio Visual (Recommendation F.721 [79])</p> <p>All the other values are reserved</p>
10	<p>Value[6] Optional bytes</p> <p>For future enhancements in protocol. Also can be used to customize the protocol with other information with this packet.</p>

0x2752 Low Layer Compatibility (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2752
2, 3	Length Variable
4, 5	Value[1] First byte of Q931 Low Layer Compatibility

0x2753 Calling Party Sub Address (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2753
2, 3	Length Variable
4	Value[1] Starts from the octet 2 of the calling party sub address of <i>ITU RECMN*Q.931 93</i> .

0x2754 Called Party Sub Address (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2754
2, 3	Length Variable
4, 5	Value[1] Starts from the octet 2 of the calling party sub address of <i>ITU RECMN*Q.931 93</i> .

0x2755 Low Layer Compatibility Layer 1 Info (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2755
2, 3	Length Variable

4, 5	<p>Value[0] User information layer 1 protocol</p> <p>0x01 CCITT standardized rate adaption V.110 [7] and X.30 [8].</p> <p>0x02 Recommendation G.711 [10] Mu Law</p> <p>0x03 Recommendation G.711 A-law</p> <p>0x04 Recommendation G.721 [11] 32 K bits/s ADPCM and recommendation I.460 [15]</p> <p>0x05 Recommendations H.221 and H.242</p> <p>0x07 Non CCITT standardized rate adaption.</p> <p>0x08 CCITT standardized rate adaption V.120 [9].</p> <p>0x09 CCITT standardized rate adaption X.31 [14] HDLC flag stuffing.</p> <p>All the other values are reserved.</p>
	<p>Value[1] User Rate</p> <p>0x00 Rate is indicated by E bits specified in Recommendation I.460 or may be negotiated in-band.</p> <p>0x01 0.6 k bits/s Recommendations V.6 [16] and X.1 [17]</p> <p>0x02 1.2 k bits/s Recommendation V.6</p> <p>0x03 2.4 k bits/s Recommendations V.6 and X.1</p> <p>0x04 3.6 k bits/s Recommendation V.6</p> <p>0x05 4.8 k bits/s Recommendations V.6 and X.1</p> <p>0x06 7.2 k bits/s Recommendation V.6</p> <p>0x07 8 k bits/s Recommendations I.460</p> <p>0x08 9.6 k bits/s Recommendations V.6 and X.1</p> <p>0x09 14.4 k bits/s Recommendation V.6</p> <p>0x0a 16 k bits/s Recommendations I.460</p> <p>0x0b 19.2 k bits/s Recommendation V.6</p> <p>0x0c 32 k bits/s Recommendations I.460</p> <p>0x0e 48 k bits/s Recommendations V.6 and X.1</p> <p>0x0f 56 k bits/s Recommendation V.6</p> <p>0x10 64 k bits/s Recommendation X.1</p> <p>0x15 0.1345 k bits/s Recommendation X.1</p> <p>0x16 0.100 k bits/s Recommendation X.1</p> <p>0x17 0.075/ 1.2 k bits/s Recommendations V.6 & X.1</p> <p>0x18 1.2/0.075 k bits/s Recommendations V.6 & X.1</p> <p>0x19 0.050 k bits/s Recommendations V.6 & X.1</p> <p>0x1a 0.075 k bits/s Recommendations V.6 & X.1</p> <p>0x1b 0.110 k bits/s Recommendations V.6 & X.1</p> <p>0x1c 0.150 k bits/s Recommendations V.6 & X.1</p> <p>0x1d 0.200 k bits/s Recommendations V.6 & X.1</p> <p>0x1e 0.300 k bits/s Recommendations V.6 & X.1</p> <p>0x1f 12 k bits/s Recommendations V.6</p> <p>All the other values are reserved.</p>

	<p>Value[2] Intermediate Rate</p> <p>0x00 Not used</p> <p>0x01 8 k bits/s</p> <p>0x02 16 k bits/s</p> <p>0x03 32 k bits/s</p> <p>All the other values are reserved.</p>
	<p>Value[3] Physical properties</p> <p>Negotiation Indicator:</p> <p>0 In-band negotiation not possible.</p> <p>1 In-band negotiation possible.</p> <p>Network Independent Clock (NIC) on Transmit:</p> <p>0 Not required to send data with network independent.</p> <p>1 Required to send data with network independent clock.</p> <p>Network Independent Clock (NIC) on Receive:</p> <p>0 Cannot accept data with network independent clock (sender does not support this optional procedure).</p> <p>1 Can accept data with network independent clock (sender does support this optional procedure).</p> <p>Flow Control on Transmit:</p> <p>0 Not required to send data with flow control mechanism.</p> <p>1 Required to send data with flow control mechanism.</p> <p>Flow control on Receive:</p> <p>0 Cannot accept data with flow control mechanism (sender does not support this optional procedure).</p> <p>1 Can accept data with flow control mechanism (sender does support this optional procedure).</p> <p>Header / No Header:</p> <p>This provides rate adaption header/ no header information.</p> <p>0 Rate adaption header not included.</p> <p>1 Rate adaption header included.</p> <p>Multiple Frame establishment support in data link:</p> <p>0 Multiple frame establishments not supported. Only UI frames allowed.</p> <p>1 Multiple frame establishments supported.</p>

	<p>Value[4] Physical Mode</p> <p>Synchronous/ Asynchronous: 0 Synchronous data. 1 Asynchronous data.</p> <p>Mode of operation: 0 Bit transparent mode of operation. 1 Protocol sensitive mode of operation.</p> <p>Logical Link Identifier Negotiation: 0 Default LLI=256 only. 1 Full protocol negotiation.</p> <p>Assignor/ Assignee: 0 Message originator is "default assignee". 1 Message originator is "assignor only".</p> <p>In-band negotiation: 0 Negotiation is done with user information messages on a temporary signaling connection. 1 Negotiation is done in-band using logical link zero.</p> <p>Duplex Mode: 0 Half Duplex. 1 Full Duplex.</p>
	<p>Value[5] Number of stop bits</p> <p>0x00 Not used 0x01 1 bit 0x02 1.5 bits 0x03 2 bits</p> <p>All the other values are reserved.</p>
	<p>Value[6] Number of data bits</p> <p>0x00 Not used 0x01 5 bit 0x02 7 bits 0x03 8 bits</p> <p>All the other values are reserved.</p>
	<p>Value[7] Parity</p> <p>0x00 Odd 0x02 Even 0x03 None 0x04 Forced to 0 0x05 Forced to 1</p> <p>All the other values are reserved.</p>

0x2756 Low Layer Compatibility Layer 2 Information (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2756
2, 3	Length Variable
4, 5	Value[0,1] User information layer 2 protocol 0x01 Basic mode ISO 1745 [36] 0x02 Recommendation Q.921 (I.441) [3] 0x06 Recommendation X.25 [5], link layer 0x0f Recommendation X.25 Multi link 0x08 Extended LAP-B for half duplex operation (T.71[37]) 0x09 HDLC ARM (ISO 4335)[38] 0x0a HDLC NRM (ISO 4335) 0x0b HDLC ABM (ISO 4335) 0x0c LAN logical link control (ISO 8802/2) 0x0d Recommendation X.75 [40]. Single link procedure (SLP). 0x0e Recommendation Q.922 0x0f Core aspects of recommendations Q.922 0x10 User specified 0x11 ISO 7776 DTE-DTE operation. All the other values are reserved.
6, 7	Value[2,3] Mode Mode of Operation: 0001 Normal mode of operation. 0010 Extended mode of operation. All the other values are reserved. Q.933 Use: 0000 For use when the coding defined in recommendation Q.933 is not used. All the other values are reserved.
:	Value[4] User information layer 2 information This field is user defined. 0xFF is reserved.
:	Value[5] Window size 1-127 All other values are reserved.

0x2757 Low Layer Compatibility Layer 3 Information

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2757
2, 3	Length Variable
4	Value[1] User information layer 3 protocol 0x01 Basic mode ISO 1745 [36] 0x02 Recommendation Q.921 (I.441) [3] 0x06 Recommendation X.25 [5], link layer 0x0f Recommendation X.25 Multi link 0x08 Extended LAP-B for half duplex operation (T.71[37]) 0x09 HDLC ARM (ISO 4335)[38] 0x0a HDLC NRM (ISO 4335) 0x0b HDLC ABM (ISO 4335) 0x0c LAN logical link control (ISO 8802/2) 0x0d Recommendation X.75 [40]. Single link procedure (SLP). 0x0e Recommendation Q.922 0x0f Core aspects of recommendations Q.922 0x10 User specified 0x11 ISO 7776 DTE-DTE operation. All the other values are reserved.
5	Value[2] Mode 0x02 Recommendation Q.931 0x06 Recommendation X.25, packet layer 0x07 ISO/IEC 8208 [41] 0x08 CCITT Recommendation X.223 0x09 ISO/IEC 8473 [43] 0x0a Recommendation T.70 [32] 0x0b ISO/IEC TR 9577[82] 0x10 User specified All the other values are reserved.
	Value[3] Optional layer 3 information This field is user defined. 0xFF is reserved.

	Value[4] Default packet size 0x04 Default packet size 16 octets 0x05 Default packet size 32 octets 0x06 Default packet size 64 octets 0x07 Default packet size 128 octets 0x08 Default packet size 256 octets 0x09 Default packet size 512 octets 0x0a Default packet size 1024 octets 0x0b Default packet size 2048 octets 0x0c Default packet size 4096 octets All the other values are reserved.
	Value[5] Packet Window size 1-127 All other values are reserved.

0x2758 Pulse Notification (Interworking Only)

This message is sent with optional data. The significance is in the message type. The optional data may have any user-defined significance or there could be no optional data.

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2758
2, 3	Length Variable
:	Value[0] Optional Data
:	Value[1] Optional Data

0x2759 Call State (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2759
2, 3	Length Variable
4	Value[0] Coding Standard 0x00 CCITT Standardized Coding 0x01 ISO/IEC Standard 0x02 National Standard 0x03 Standard specific to identified location

User Side
0x0000 Null
0x0001 Call Initiated
0x0002 Overlap sending
0x0003 Outgoing call proceeding
0x0004 Call delivered
0x0005 Call present
0x0006 Call Received
0x0007 Connect Request
0x0008 Incoming call proceeding
0x0009 Active
0x000a Disconnect request
0x000b Disconnect Indication
0x000c Suspend request
0x000d Resume request
0x000e Overlap receive
0x000f Restart Request
0x0010 Restart
0x0011 Call abort
Network Side
0x0100 Null
0x0101 Call Initiated
0x0102 Overlap sending
0x0103 Outgoing call proceeding
0x0104 Call delivered
0x0105 Call present
0x0106 Call Received
0x0107 Connect Request
0x0108 Incoming call proceeding
0x0109 Active
0x010a Disconnect request
0x010b Disconnect Indication
0x010c Suspend request
0x010d Resume request
0x010e Overlap receive
0x010f Restart Request
0x0110 Restart
0x0111 Call abort
All other values are reserved.

0x275A Generic Command (Interworking Only)

Use this TLV to send a generic command to the network entity to perform some predefined command codes.

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x275A
2, 3	Length Variable
4, 5	Value[0,1]Generic Command Length MSB, LSB
6, 7	Value[2,3] Generic Command MSB, LSB
:	:
:	Value[n] Generic Command MSB, LSB

0x275B Generic Command Response (Interworking Only)

Use this TLV to respond to generic command. The response code should be same as the command code to confirm the command (though not required).

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x275B
2, 3	Length Variable
4, 5	Value[0,1]Generic Command Response Length MSB, LSB
6, 7	Value[2,3] Generic Command Response MSB, LSB
:	:
:	Value[n] Generic Command Response MSB, LSB

7	<p>Value[1] Pulse Type</p> <p>0x7f Pulsed normal polarity 0x7d Pulsed reversed polarity 0x7c Pulsed battery on c-wire 0x7b Pulsed on hook 0x7a Pulsed reduced battery 0x79 Pulsed no battery 0x78 Initial ring 0x77 Meter pulse 0x76 50 KHz Pulse 0x75 Register recall (timed loop open) 0x74 Pulsed off hook 0x73 Pulsed b wire connection to earth 0x72 Earth loop pulse 0x71 Pulsed b wire connected to battery 0x70 Pulsed a wire connected to battery 0x6f Pulsed c wire connection to earth 0x6e Pulsed c wire disconnected 0x6d Pulsed normal battery 0x6c Pulsed a wire disconnected 0x6b Pulsed b wire disconnected 0x50 Disable Autonomous response (defined here also for illustration).</p> <p>This value must be unique and should not be repeated in the steady signal's signal type field.</p>
:	<p>Value[2] Suppression indication</p> <p>0x00 No suppression. 0x01 Suppression allowed by pre-defined V5.1 SIGNAL message from LE 0x02S Suppression allowed by pre-defined V5.1 message from TE 0x03 Suppression allowed by pre-defined V5.1 SIGNAL message from LE or predefined line signal from TE</p> <p>All other values are reserved.</p>
:	<p>Value[3] Pulse Duration Type</p> <p>A predefined description, which consists of the time for the pulse in total and duty cycle.</p>

	<p>Value[4] Acknowledgement Request Indicator</p> <p>0x00 No acknowledge requested</p> <p>0x01 Ending acknowledgement requested when finished each pulse</p> <p>0x02 Ending acknowledgement requested when finished all pulse</p> <p>0x03 Start of pulse acknowledgement requested All other values are reserved.</p>
	<p>Value[5] Number of Pulses The number of pulses sent.</p>
	<p>Value[6] Response Used to enable/disable autonomous acknowledge message.</p> <p>Use values as defined in Pulsed Type (in this TLV) and Steady Signal Type (in Steady-Signal TLV 0x275E).</p> <p>The following values are also valid:</p> <p>0x50 Disable autonomous response 0xFF Response not used</p>

**0x275E Steady Signal
(Interworking Only)**

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x275E
2, 3	Length Variable
5	<p>Value[0] Steady Signal Type</p> <p>0x00 Normal polarity 0x01 Reversed polarity 0x02 Battery on C wire 0x03 No battery on C wire 0x04 Off hook (loop closed) 0x05 On hook (loop open) 0x06 Battery on A wire 0x07 A wire on earth 0x08 No battery on A wire 0x09 No battery on B wire 0x0a Reduced battery 0x0b No battery 0x0c Alternate reduced power/ no power 0x0d Normal battery 0x0e Stop ringing 0x0f Start pilot frequency 0x10 Stop pilot frequency 0x11 Low impedance on B wire 0x12 B wire connected to earth 0x13 B wire disconnected from earth 0x14 Battery on B wire 0x15 Low loop impedance 0x16 High loop impedance 0x17 Anomalous loop impedance 0x18 A wire disconnected from earth 0x19 C wire on earth 0x1a C wire disconnected from earth 0x50 Disable Autonomous response</p> <p>Before adding the new value, this value should be unique and should not be repeated in the Pulsed Signal's Signal Type field.</p> <p>All the other values are reserved.</p>
	<p>Value[1] Response</p> <p>Used to enable/disable autonomous acknowledge message. Use values as defined in Pulsed Type (in this TLV) and Steady Signal Type (in Steady-Signal TLV 0x275E).</p> <p>The following values are also valid:</p> <p>0x50 Disable autonomous response 0xFF Response not used</p>

**0x275F Recognition Time
(Interworking Only)**

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x275F
2, 3	Length Variable
4	Value[0] Signal Type Use values as defined in Pulsed Type (in Pulsed Signal TLV 0x275D) and Steady Signal Type (in Steady-Signal TLV 0x275E).
5	Value[1] Duration Type Predefined description that is the actual value of the recognition time.

**0x2760 Line Information
(Interworking Only)**

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2760
2, 3	Length Variable
:	Value[0] Parameter 0x00 Impedance marker reset 0x01 Impedance marker set 0x02 Low loop impedance 0x03 Anomalous loop impedance All other values are reserved.

0x2761 Channel Identification

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2761
2, 3	Length Variable
4,5	Value[0] Address Type 0x0000 Span Channel 0x0001 V5 Interface
6-n	Value[1] Address Data for Span Channel Address Data 1 0 (reserved) Address Data 2 0 (reserved) Address Data 3 Span MSB Address Data 4 Span LSB Address Data 5 Channel Address Data 6 0 (reserved) Rest of the values are reserved. Address Data for V5 Interface Address Data 1 Node ID MSB Address Data 2 Node ID LSB Address Data 3 V5 Interface ID MSB Address Data 4 V5 Interface ID LSB Address Data 5 User Port MSB Address Data 6 User Port LSB Rest of the values are reserved.
:	Value[2] Optional Bytes

**0x2762 Date and Time
(Interworking Only)**

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2762
2, 3	Length Variable
4, 5	Value[0] Year AD MSB, LSB
6	Value[1] Month (1-12) 1- January : 12 - December
7	Value[2] Day 1-31

8	Value[3] Hour 0-24
9	Value[4] Minute 0-59
10	Value[5] Seconds 0-59
11, 12	Value[6] Millisecond 0-999
13	<p>Value[7] Zone Information</p> <p>0x00 GMT 0x01 EST (USA & Canada) 0x02 CST (USA & Canada) 0x03 MST (USA & Canada) 0x04 PST (USA & Canada) 0x05 Atlantic Time (Canada) 0x06 Newfoundland Time (Canada) 0x07 Indian Time 0xF0 to 0xFD User Defined 0xFF Not set or Unknown</p> <p>All the other values are reserved.</p>
14	<p>Value[8] Daylight Savings Considerations</p> <p>0x00 DST not in effect 0x01 DST in effect 0x02 DST not used at all in this time zone. 0xF0 to 0xFD User Defined 0xFF Not set or Unknown</p> <p>All the other values are reserved.</p>

0x2763 Carrier Information (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2763
2, 3	Length Variable
4	<p>Value[0] Type of network and network ID plan</p> <p>Bit 7 Spare bit, should always be 0</p> <p>Bits 6-4 are for type of network ID:</p> <p>010 national network identification 000 CCITT-standardized identification 111 not sure about whether it is national or CCITT</p> <p>Note that CarrierID parameter is just for ANSI national network (ANSI T1.113-1995, 3.8A), so it should only be set as "010". But the current implementation of IW will translate TNS into universal Carrier ID as well, which may be used in CCITT, so we have to keep "000" and "111" in our usage.</p> <p>Bits 3-0 Network ID plan</p> <p>0001 3-digit carrier ID 0010 4-digit carrier ID 0000 Carrier ID parameter does not present in the incoming call, that the carrier ID unknown, and this whole parameter is just a void Carrier ID.</p>
5	Value[1] Circuit number
6	Value[2] Number of Digits
7	Value[3] BCD digits pair
8	Value[4] BCD digits pair

0x277D Universal Message Envelope (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x277D
2, 3	Length Variable
4, 5	Value [0] Number of IEs
6, 7	Value [1] IE Type

8, 9	Value [2] IE Length
10, 11	Value [3] IE Data
:	:
:	Value [n] IE Type
:	Value [n] IE Length
:	Value [n] IE Data

0x277E Remote Side Protocol

Use this TLV to specify the protocol being used. This TLV can be used in Interworking and Standard environments.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Outseize Control message

Route Control message

Byte	Description
0, 1	Tag: Remote Side Protocol (0x277E)
2, 3	Length: 0x0003
4	Protocol Type 0x01 SS7 0x02 ISDN 0x03 T1 0x04 E1 0x05 Universal 0x06 V5.1 0x07 V5.2 0x08 SIP 0x09 H.323
5	Protocol Subtype 0x00 Reserved
6	Subscription 0x00 Reserved

0x277F - 0x2783 Display (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2783 - 0x2788
2, 3	Length Variable
4, 5	Value[0] Display Length No length restriction. Each process sets their own length limit and truncates excess data.
:	Value[1] Display Information (IA5/ASCII characters)

0x2784 Charge Number (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2784
2, 3	Length Variable
4	Value[0] Nature of address indicator
5	Value[1] Numbering plan identifier
6	Value[2] Number of digits
10	Value[3] BCD Digit Pair 1
11	Value[4] BCD Digit Pair 2
:	:

0x278C Connected Number Subaddress (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Refer to *ITU Q.951/A.731 '93*.

0x278F Originating Line Information (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x278F
2, 3	Length Variable
4	Value[0] Number of digits
11	Value[1] BCD digit pair For the complete listing of values, see the North American Numbering Plan Administrator web site: http://www.nanpa.com/number_resource_info/ani_ii_assignments.html
:	:

0x2792 Source IP Address

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

Outseize Control message

Byte	Description
0, 1	Tag 0x2792 Source IP Address
2, 3	Length 0x0004
4-7	Value[0-3] IP Address

0x2793 Source RTP Port

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

Outseize Control message

Byte	Description
0, 1	Tag 0x2793 Source IP Port
2, 3	Length 0x0004
4, 5	Value[0, 1] 0x0000 (Do not change)
6, 7	Value[2, 3] RTP Port

0x2794 Destination IP Address

Used in:

NPDI Universal ICB (0x0033) in:

Route Control message

Request for Service with Data message

Outseize Control message

Byte	Description
0, 1	Tag 0x2794 Destination IP Address
2, 3	Length 0x0004
4-7	Value[0-3] IP Address

0x2795 Destination RTP Port

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Route Control message

Outseize Control message

Byte	Description
0, 1	Tag 0x2795 Destination RTP Port
2, 3	Length 0x0004
4, 5	Value[0, 1] 0x0000 (Do not change)
6, 7	Value[2, 3] RTP Port

0x27AF NPDI SIP Fax Mode

Use this TLV in the following messages:

- *Outseize Control* (0x002C) and *Route Control* (0x00E8)
Host managed outseize control and route control. Use as a mandatory TLV in this message.
- *PPL Table Download* (0x00D6)
Use as a L4-Router TLV in this message.

- *Route Control (0x00E8)*
L4 managed route control. Use as an optional TLV in this message.

Byte	Description
0,1	Tag 0x27AF NPDI SIP Fax Mode
2,3	Length 0x0001
4	Value[0] 00=Disabled 01= Initiate T.38 re-INVITE 02= Initiate Bypass (PCM) re-INVITE

0x27B0 RTP Payload Type

Use this TLV in the NPDI Universal ICB (0x0033) to configure the RTP Payload Type.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Route Control message

Outseize Control message

Important! Do not confuse this TLV with 0x0100, which is the regular resource attribute TLV for RTP Payload Type. TLV 0x27B0 is the RTP Payload Type in NPDI format.

Byte	Description
0, 1	Tag 0x27B0 RTP Payload Type
2, 3	Length 0x0002
4	Value[0,1] Direction 0x00 - Ingress 0x01 - Egress (byte 5 on next page)

5	<p>Value: Payload Type</p> <p><u>VDAC-ONE and IP Network Interface Series 2 Card</u></p> <table> <tr><td>0x00</td><td>G.711</td><td>A-Law</td></tr> <tr><td>0x01</td><td>G.711</td><td>μ-Law (Default)</td></tr> <tr><td>0x02</td><td>G.726</td><td>16 Kbps</td></tr> <tr><td>0x03</td><td>G.726</td><td>24 Kbps</td></tr> <tr><td>0x04</td><td>G.726</td><td>32 Kbps</td></tr> <tr><td>0x05</td><td>G.726</td><td>40 Kbps</td></tr> <tr><td>0x06</td><td>G.727</td><td>16 Kbps *</td></tr> <tr><td>0x07</td><td>G.727</td><td>24, 16 Kbps *</td></tr> <tr><td>0x08</td><td>G.727</td><td>24 Kbps *</td></tr> <tr><td>0x09</td><td>G.727</td><td>32, 16 Kbps *</td></tr> <tr><td>0x0A</td><td>G.727</td><td>32, 24 Kbps *</td></tr> <tr><td>0x0B</td><td>G.727</td><td>32 Kbps *</td></tr> <tr><td>0x0C</td><td>G.727</td><td>40, 16 Kbps *</td></tr> <tr><td>0x0D</td><td>G.727</td><td>40, 24 Kbps *</td></tr> <tr><td>0x0E</td><td>G.727</td><td>40, 32 Kbps *</td></tr> <tr><td>0x0F</td><td>G.723</td><td>5.3 Kbps</td></tr> <tr><td>0x10</td><td>G.723</td><td>6.3 Kbps</td></tr> <tr><td>0x11</td><td>G.729</td><td></td></tr> </table> <p><u>SIP/H.323/CallAgent</u></p> <p>Codec must also be supported by the VDAC-ONE or IP Network Interface Series 2 card for bearer calls.</p> <table> <tr><td>0x01</td><td>G.711</td><td>A-Law</td></tr> <tr><td>0x02</td><td>G.711</td><td>μ-Law (Default)</td></tr> <tr><td>0x03</td><td>G.726</td><td>16 Kbps</td></tr> <tr><td>0x04</td><td>G.726</td><td>24 Kbps</td></tr> <tr><td>0x05</td><td>G.726</td><td>32 Kbps</td></tr> <tr><td>0x06</td><td>G.726</td><td>40 Kbps</td></tr> <tr><td>0x07</td><td>G.727</td><td>16 Kbps *</td></tr> <tr><td>0x08</td><td>G.727</td><td>24, 16 Kbps *</td></tr> <tr><td>0x09</td><td>G.727</td><td>24 Kbps *</td></tr> <tr><td>0x0A</td><td>G.727</td><td>32, 16 Kbps *</td></tr> <tr><td>0x0B</td><td>G.727</td><td>32, 24 Kbps *</td></tr> <tr><td>0x0C</td><td>G.727</td><td>32 Kbps *</td></tr> <tr><td>0x0D</td><td>G.727</td><td>40, 16 Kbps *</td></tr> <tr><td>0x0E</td><td>G.727</td><td>40, 24 Kbps *</td></tr> <tr><td>0x0F</td><td>G.727</td><td>40, 32 Kbps *</td></tr> <tr><td>0x10</td><td>G.723</td><td>5.3 Kbps</td></tr> <tr><td>0x11</td><td>G.723</td><td>6.3 Kbps</td></tr> <tr><td>0x12</td><td>G.729</td><td></td></tr> <tr><td>0x13</td><td>G.728</td><td></td></tr> <tr><td>0x14</td><td>1016</td><td></td></tr> <tr><td>0x15</td><td>G.721</td><td></td></tr> <tr><td>0x16</td><td>GSM</td><td></td></tr> <tr><td>0x17</td><td>DV14</td><td>8 Kbps</td></tr> <tr><td>0x18</td><td>DV14</td><td>11 Kbps</td></tr> <tr><td>0x19</td><td>DV14</td><td>16 Kbps</td></tr> <tr><td>0x1A</td><td>DV14</td><td>22 Kbps</td></tr> <tr><td>0x1B</td><td>LPC</td><td></td></tr> <tr><td>0x1C</td><td>G722</td><td>64 Kbps</td></tr> <tr><td>0x1D</td><td>L16</td><td>1CH</td></tr> <tr><td>0x1E</td><td>L16</td><td>1CH</td></tr> <tr><td>0x1F</td><td>MPA</td><td></td></tr> <tr><td>0x20</td><td>CELB</td><td></td></tr> <tr><td>0x21</td><td>JPEG</td><td></td></tr> <tr><td>0x22</td><td>NV</td><td></td></tr> <tr><td>0x23</td><td>H261</td><td></td></tr> <tr><td>0x24</td><td>MPV</td><td></td></tr> <tr><td>0x25</td><td>MP2T</td><td></td></tr> <tr><td>0x26</td><td>H263</td><td></td></tr> <tr><td>0x27</td><td>G711A</td><td>56 Kbps</td></tr> <tr><td>0x28</td><td>G711U</td><td>56 Kbps</td></tr> <tr><td>0x29</td><td>G.722</td><td>56 Kbps</td></tr> <tr><td>0x2A</td><td>G.722</td><td>48 Kbps</td></tr> </table> <p>* VDAC-ONE card only</p>	0x00	G.711	A-Law	0x01	G.711	μ-Law (Default)	0x02	G.726	16 Kbps	0x03	G.726	24 Kbps	0x04	G.726	32 Kbps	0x05	G.726	40 Kbps	0x06	G.727	16 Kbps *	0x07	G.727	24, 16 Kbps *	0x08	G.727	24 Kbps *	0x09	G.727	32, 16 Kbps *	0x0A	G.727	32, 24 Kbps *	0x0B	G.727	32 Kbps *	0x0C	G.727	40, 16 Kbps *	0x0D	G.727	40, 24 Kbps *	0x0E	G.727	40, 32 Kbps *	0x0F	G.723	5.3 Kbps	0x10	G.723	6.3 Kbps	0x11	G.729		0x01	G.711	A-Law	0x02	G.711	μ-Law (Default)	0x03	G.726	16 Kbps	0x04	G.726	24 Kbps	0x05	G.726	32 Kbps	0x06	G.726	40 Kbps	0x07	G.727	16 Kbps *	0x08	G.727	24, 16 Kbps *	0x09	G.727	24 Kbps *	0x0A	G.727	32, 16 Kbps *	0x0B	G.727	32, 24 Kbps *	0x0C	G.727	32 Kbps *	0x0D	G.727	40, 16 Kbps *	0x0E	G.727	40, 24 Kbps *	0x0F	G.727	40, 32 Kbps *	0x10	G.723	5.3 Kbps	0x11	G.723	6.3 Kbps	0x12	G.729		0x13	G.728		0x14	1016		0x15	G.721		0x16	GSM		0x17	DV14	8 Kbps	0x18	DV14	11 Kbps	0x19	DV14	16 Kbps	0x1A	DV14	22 Kbps	0x1B	LPC		0x1C	G722	64 Kbps	0x1D	L16	1CH	0x1E	L16	1CH	0x1F	MPA		0x20	CELB		0x21	JPEG		0x22	NV		0x23	H261		0x24	MPV		0x25	MP2T		0x26	H263		0x27	G711A	56 Kbps	0x28	G711U	56 Kbps	0x29	G.722	56 Kbps	0x2A	G.722	48 Kbps
0x00	G.711	A-Law																																																																																																																																																																																			
0x01	G.711	μ-Law (Default)																																																																																																																																																																																			
0x02	G.726	16 Kbps																																																																																																																																																																																			
0x03	G.726	24 Kbps																																																																																																																																																																																			
0x04	G.726	32 Kbps																																																																																																																																																																																			
0x05	G.726	40 Kbps																																																																																																																																																																																			
0x06	G.727	16 Kbps *																																																																																																																																																																																			
0x07	G.727	24, 16 Kbps *																																																																																																																																																																																			
0x08	G.727	24 Kbps *																																																																																																																																																																																			
0x09	G.727	32, 16 Kbps *																																																																																																																																																																																			
0x0A	G.727	32, 24 Kbps *																																																																																																																																																																																			
0x0B	G.727	32 Kbps *																																																																																																																																																																																			
0x0C	G.727	40, 16 Kbps *																																																																																																																																																																																			
0x0D	G.727	40, 24 Kbps *																																																																																																																																																																																			
0x0E	G.727	40, 32 Kbps *																																																																																																																																																																																			
0x0F	G.723	5.3 Kbps																																																																																																																																																																																			
0x10	G.723	6.3 Kbps																																																																																																																																																																																			
0x11	G.729																																																																																																																																																																																				
0x01	G.711	A-Law																																																																																																																																																																																			
0x02	G.711	μ-Law (Default)																																																																																																																																																																																			
0x03	G.726	16 Kbps																																																																																																																																																																																			
0x04	G.726	24 Kbps																																																																																																																																																																																			
0x05	G.726	32 Kbps																																																																																																																																																																																			
0x06	G.726	40 Kbps																																																																																																																																																																																			
0x07	G.727	16 Kbps *																																																																																																																																																																																			
0x08	G.727	24, 16 Kbps *																																																																																																																																																																																			
0x09	G.727	24 Kbps *																																																																																																																																																																																			
0x0A	G.727	32, 16 Kbps *																																																																																																																																																																																			
0x0B	G.727	32, 24 Kbps *																																																																																																																																																																																			
0x0C	G.727	32 Kbps *																																																																																																																																																																																			
0x0D	G.727	40, 16 Kbps *																																																																																																																																																																																			
0x0E	G.727	40, 24 Kbps *																																																																																																																																																																																			
0x0F	G.727	40, 32 Kbps *																																																																																																																																																																																			
0x10	G.723	5.3 Kbps																																																																																																																																																																																			
0x11	G.723	6.3 Kbps																																																																																																																																																																																			
0x12	G.729																																																																																																																																																																																				
0x13	G.728																																																																																																																																																																																				
0x14	1016																																																																																																																																																																																				
0x15	G.721																																																																																																																																																																																				
0x16	GSM																																																																																																																																																																																				
0x17	DV14	8 Kbps																																																																																																																																																																																			
0x18	DV14	11 Kbps																																																																																																																																																																																			
0x19	DV14	16 Kbps																																																																																																																																																																																			
0x1A	DV14	22 Kbps																																																																																																																																																																																			
0x1B	LPC																																																																																																																																																																																				
0x1C	G722	64 Kbps																																																																																																																																																																																			
0x1D	L16	1CH																																																																																																																																																																																			
0x1E	L16	1CH																																																																																																																																																																																			
0x1F	MPA																																																																																																																																																																																				
0x20	CELB																																																																																																																																																																																				
0x21	JPEG																																																																																																																																																																																				
0x22	NV																																																																																																																																																																																				
0x23	H261																																																																																																																																																																																				
0x24	MPV																																																																																																																																																																																				
0x25	MP2T																																																																																																																																																																																				
0x26	H263																																																																																																																																																																																				
0x27	G711A	56 Kbps																																																																																																																																																																																			
0x28	G711U	56 Kbps																																																																																																																																																																																			
0x29	G.722	56 Kbps																																																																																																																																																																																			
0x2A	G.722	48 Kbps																																																																																																																																																																																			

0x27B1 RTP Payload Size

Use this TLV in the NPDI Universal ICB (0x0033) to configure the RTP Payload Size.

By changing the multiple of the payload size, you can configure how much encoded PCM data a packet contains.

Example: For G.711 on the VDAC-ONE card, a 160-byte payload of encoded PCM data is sent at the G.711 Basic Packet Rate of every 20 milliseconds. But if you change the multiple for G.711 to 2x, then a 320-byte payload is sent every 40 milliseconds.

In effect, changing the multiple also changes the throughput available over an Ethernet network. Larger packets create greater network delay overall, but they use the Ethernet bandwidth more efficiently. Smaller packets may use the Ethernet bandwidth less efficiently, but they may provide better voice quality.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Route Control message

Outseize Control message

Important! Do not confuse this TLV with 0x0101 which is the regular resource attribute TLV for RTP Payload Size. Tag 0x27B1 is the RTP Payload Size in NPDI format.

Byte	Description
0, 1	Tag: RTP Payload Size (0x27B1)
2, 3	Length: 0x0002
4	Value [0] Direction 0x00 Ingress 0x01 Egress
5	Value [1]: Multiple of the Basic Packet Rate (Default = 0x01)

0x27B2 NPDI RTP Silence Suppression

The RTP Silence Suppression TLV block enables and disables silence suppression.

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

Outseize Control message

Do not confuse this TLV with Tag 0x0102 which is the regular resource attribute TLV for Silence Suppression. Tag 0x27B2 is the RTP Silence Suppression in NPDI format.

Byte	Description
0, 1	Tag NPDI RTP Silence Suppression (0x27B2)
2, 3	Length 0x0001
4	Value[0] 0x00 Disable (Default) 0x01 Enable

0x27B3 NPDI RTP Connection Mode

This TLV block allows you to set the connection mode. A line can be connected to receive only, transmit only, or both.

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

Outseize Control message

Do not confuse this TLV with TLV 0x01DB, which is the regular resource attribute TLV for RTP Connection Mode. Tag 0x27B3 is the RTP Connection Mode in NPDI format.

Byte	Description
0, 1	Tag: NPDI RTP Connection Mode (0x27B3)
2, 3	Length: 0x0001
4	Value[0] 0x00 Two-way connection 0x01 One-way listen/receive only 0x02 One-way talk/transmit only 0x03 Hold Mode

0x27B4 IP Address Criteria

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

Byte	Description
0, 1	Tag IP Address Criteria (0x27B4)
2, 3	Length 0x0004
4. . .	Value IP Address

0x27C1 IP Signaling Series 3 Card ID

Used in:

0x0033 NPDI Universal ICB in:

Outsize Control message

Route Control message

Byte	Description
0, 1	Tag IP Signaling Series 3 card (0x27C1)
2, 3	Length 0x0002
4, 5	Value 0x0000 - 0x003F

0x27C2 Remote End Signaling IP Address (Q.931)

This TLV includes the IP address of the remote end with respect to the IP Signaling card. For inbound calls, this IP address is for the originator of the call. For outbound calls, this IP address is for the destination end point.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service message

Route Control message

Byte	Description
0, 1	Tag Remote End Signaling IP Address (0x27C2)
2, 3	Length 0x0004
4-7	Value IP Address

0x27C3 Remote End Signaling Port (Q.931)

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

Byte	Description
0, 1	Tag Remote End Signaling Port (0x27C3)
2, 3	Length Variable
4 . . .	Value Variable If the Remote End Signaling Port is not specified, the default value is 1720.

0x27C4 Source IP

Used in:
 0x0033 NPDI Universal ICB in:
Request for Service message
Route Control message

Byte	Description
0, 1	Tag Source IP (0x27C4)
2, 3	Length 0x04
4. . .	Value Variable

0x27C5 Source Port

Used in:
 0x0033 NPDI Universal ICB in:
Request for Service message
Route Control message

Byte	Description
0, 1	Tag Source Port (0x27C5)
2, 3	Length Variable
4. . .	Value Variable

0x27C9 Charge Area Information Parameter

Used in:
 0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x27C9
2, 3	Length Variable
4	Value[1] Number of Digits
5	Value[2] First pair of CBD digits
6	Value[3] Second pair of BCD digits
7	Value[3] Third pair of BCD digits

0x27CA Carrier Information Parameter (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x27CA
2, 3	Length Variable
4	Value[1] Carrier Information Name 11111011 Calling Carrier Information 11111100 Called Carrier Information
5	Value[2] Carrier Information Parameter 11111110 Carrier Identification Code
6	Value[3] Number of Digits
7	Value[4] First pair of CBD digits
8	Value[5] Second pair of BCD digits
9	Value[6] Third pair of BCD digits
10	Remainder of Raw SS7 Parameters Not currently used.

0x27CB Non-Presentation Reason (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x27CB
2, 3	Length Variable
4	Value[1] Non-presentation Reason This parameter is required for KDDI and the CLIP/CLIR service. 0x01 Unnoticed because of user rejection 0x02 Unnoticed because of conflicting services 0x03 Unnoticed because of public telephone call

0x27D0 Partial Calling Line Identity (Interworking Only)

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x27D0
2, 3	Length Variable
4	Value[0] Type of Switch (BCD) Bits 1-4 (tens digit) Bits 5-8 (units digit)
5	Value[1] PNO Identity (BCD) Bits 1-4 (hundreds digit) Bits 5-8 (tens digit)
6	Value[2] Bits 1-4 PNO identity (BCD) (units digit) Bits 5-8 Switch Number (BCD) (hundreds digit)
7	Value[3] Switch Number (BCD) Bits 1-4 (tens digit) Bits 5-7 (units digit)

0x27D8 Source H.323 ID

Used in:

0x0033 NPDI Universal ICB in:

Request for Service message

Route Control message

Byte	Description
0, 1	Tag Source H.323 ID (0x27D8)
2, 3	Length Variable
4. . .	Value Source H.323 ID (IA5/ASCII NULL terminated characters.)

0x27D9 Source URL

Used in:

0x0033 NPDI Universal ICB in:

Request for Service message

Route Control message

Byte	Description
0, 1	Tag Source H.323 ID (0x27D9)
2, 3	Length Variable
4. . .	Value Variable

0x27DA Source E-mail

Used in:

0x0033 NPDI Universal ICB in:

Request for Service message*Route Control* message

Byte	Description
0, 1	Tag Source Email (0x27DA)
2, 3	Length Variable
4. . .	Value Variable

0x27DC Remote H.323 ID

Used in:

0x0033 NPDI Universal ICB in:

Request for Service message*Route Control* message

Byte	Description
0, 1	Tag Remote H.323 ID (0x27DC)
2, 3	Length Variable
4. . .	Value Variable

0x27DD Remote URL

Used in:

0x0033 NPDI Universal ICB in:

Request for Service message*Route Control* message

Byte	Description
0, 1	Tag Remote URL (0x27DD)
2, 3	Length Variable
4. . .	Value Variable

0x27DE Remote E-mail

Used in:

0x0033 NPDI Universal ICB in:

Request for Service message*Route Control* message

Byte	Description
0, 1	Tag Remote Email (0x27DE)
2, 3	Length Variable
4. . .	Value Variable

0x27E3 Release Cause Code

The following release cause codes are used for H.323 calls.

Used in:

0x0033 NPDI Universal ICB in:

Channel Released message

Channel Released with Data message

Byte	Description																																																																																												
0, 1	Tag Release Cause Code (0x27E3)																																																																																												
2, 3	Length 0x02																																																																																												
4,5	Value[0] <table border="0" style="margin-left: 20px;"> <tr> <td style="vertical-align: top;">MSB</td> <td>0x01 Release Reason</td> </tr> <tr> <td style="vertical-align: top;">LSB</td> <td>0x01 Called Party Not Registered</td> </tr> <tr> <td></td> <td>0x02 Invalid Permission Chosen</td> </tr> <tr> <td></td> <td>0x03 Request Denied</td> </tr> <tr> <td></td> <td>0x04 Undefined Reason</td> </tr> <tr> <td></td> <td>0x05 Caller Not Registered</td> </tr> <tr> <td></td> <td>0x06 Route Call to Gatekeeper</td> </tr> <tr> <td></td> <td>0x07 Invalid Endpoint Identifier Chosen</td> </tr> <tr> <td></td> <td>0x08 Resource Unavailable Chosen</td> </tr> <tr> <td></td> <td>0x09 Security Denial Chosen</td> </tr> <tr> <td></td> <td>0x0A Control Not Supported Chosen</td> </tr> <tr> <td></td> <td>0x0B Complete Address Chosen</td> </tr> <tr> <td></td> <td>0x0C Release Complete Undefined Reason Chosen</td> </tr> <tr> <td style="vertical-align: top;">MSB</td> <td>0x02 Release Reason Remote End</td> </tr> <tr> <td style="vertical-align: top;">LSB</td> <td>0x01 No Bandwidth Chosen</td> </tr> <tr> <td></td> <td>0x02 Gatekeeper</td> </tr> <tr> <td></td> <td>0x03 Unreachable Destination Rejection Chosen</td> </tr> <tr> <td></td> <td>0x04 Destination Rejection Chosen</td> </tr> <tr> <td></td> <td>0x05 Release Complete Invalid Revision Chosen</td> </tr> <tr> <td></td> <td>0x06 No Permission Chosen</td> </tr> <tr> <td></td> <td>0x07 Unreachable Gatekeeper Chosen</td> </tr> <tr> <td></td> <td>0x08 Gateway Resources Chosen</td> </tr> <tr> <td></td> <td>0x09 Bad Format Address Chosen</td> </tr> <tr> <td></td> <td>0x0A Adaptive Busy Chosen</td> </tr> <tr> <td></td> <td>0x0B In Conf Chosen</td> </tr> <tr> <td></td> <td>0x0C Release Complete Undefined Reason Chosen</td> </tr> <tr> <td></td> <td>0x0D Facility Call Deflection</td> </tr> <tr> <td></td> <td>0x0E Security Denied Chosen</td> </tr> <tr> <td></td> <td>0x0F Release Complete Called Party Not Registered Chosen</td> </tr> <tr> <td></td> <td>0x10 Release Complete Caller Not Registered Chosen</td> </tr> <tr> <td style="vertical-align: top;">MSB</td> <td>0x03 Release Reason Others</td> </tr> <tr> <td style="vertical-align: top;">LSB</td> <td>0x01 Outsize Error</td> </tr> <tr> <td></td> <td>0x02 Protocol Error</td> </tr> <tr> <td></td> <td>0x03 Socket Open Fail</td> </tr> <tr> <td></td> <td>0x04 Gatekeeper not Configured</td> </tr> <tr> <td></td> <td>0x05 ARQ Fail</td> </tr> <tr> <td></td> <td>0x06 Purge</td> </tr> <tr> <td></td> <td>0x07 Local End Release</td> </tr> <tr> <td></td> <td> MSB 0x04 Q931 Cause Location USER</td> </tr> <tr> <td></td> <td>LSB Q931 Cause code</td> </tr> <tr> <td></td> <td> MSB 0x05 Q931 Cause Location Private Network serving Local user</td> </tr> <tr> <td></td> <td>LSB Q931 Cause code</td> </tr> <tr> <td></td> <td> MSB 0x06 Q931 Cause Location Public Network serving Local user</td> </tr> <tr> <td></td> <td>LSB Q931 Cause code</td> </tr> <tr> <td></td> <td> MSB 0x07 Q931 Cause Location Transit Network</td> </tr> <tr> <td></td> <td>LSB Q931 Cause code</td> </tr> </table>	MSB	0x01 Release Reason	LSB	0x01 Called Party Not Registered		0x02 Invalid Permission Chosen		0x03 Request Denied		0x04 Undefined Reason		0x05 Caller Not Registered		0x06 Route Call to Gatekeeper		0x07 Invalid Endpoint Identifier Chosen		0x08 Resource Unavailable Chosen		0x09 Security Denial Chosen		0x0A Control Not Supported Chosen		0x0B Complete Address Chosen		0x0C Release Complete Undefined Reason Chosen	MSB	0x02 Release Reason Remote End	LSB	0x01 No Bandwidth Chosen		0x02 Gatekeeper		0x03 Unreachable Destination Rejection Chosen		0x04 Destination Rejection Chosen		0x05 Release Complete Invalid Revision Chosen		0x06 No Permission Chosen		0x07 Unreachable Gatekeeper Chosen		0x08 Gateway Resources Chosen		0x09 Bad Format Address Chosen		0x0A Adaptive Busy Chosen		0x0B In Conf Chosen		0x0C Release Complete Undefined Reason Chosen		0x0D Facility Call Deflection		0x0E Security Denied Chosen		0x0F Release Complete Called Party Not Registered Chosen		0x10 Release Complete Caller Not Registered Chosen	MSB	0x03 Release Reason Others	LSB	0x01 Outsize Error		0x02 Protocol Error		0x03 Socket Open Fail		0x04 Gatekeeper not Configured		0x05 ARQ Fail		0x06 Purge		0x07 Local End Release		 MSB 0x04 Q931 Cause Location USER		LSB Q931 Cause code		 MSB 0x05 Q931 Cause Location Private Network serving Local user		LSB Q931 Cause code		 MSB 0x06 Q931 Cause Location Public Network serving Local user		LSB Q931 Cause code		 MSB 0x07 Q931 Cause Location Transit Network		LSB Q931 Cause code
MSB	0x01 Release Reason																																																																																												
LSB	0x01 Called Party Not Registered																																																																																												
	0x02 Invalid Permission Chosen																																																																																												
	0x03 Request Denied																																																																																												
	0x04 Undefined Reason																																																																																												
	0x05 Caller Not Registered																																																																																												
	0x06 Route Call to Gatekeeper																																																																																												
	0x07 Invalid Endpoint Identifier Chosen																																																																																												
	0x08 Resource Unavailable Chosen																																																																																												
	0x09 Security Denial Chosen																																																																																												
	0x0A Control Not Supported Chosen																																																																																												
	0x0B Complete Address Chosen																																																																																												
	0x0C Release Complete Undefined Reason Chosen																																																																																												
MSB	0x02 Release Reason Remote End																																																																																												
LSB	0x01 No Bandwidth Chosen																																																																																												
	0x02 Gatekeeper																																																																																												
	0x03 Unreachable Destination Rejection Chosen																																																																																												
	0x04 Destination Rejection Chosen																																																																																												
	0x05 Release Complete Invalid Revision Chosen																																																																																												
	0x06 No Permission Chosen																																																																																												
	0x07 Unreachable Gatekeeper Chosen																																																																																												
	0x08 Gateway Resources Chosen																																																																																												
	0x09 Bad Format Address Chosen																																																																																												
	0x0A Adaptive Busy Chosen																																																																																												
	0x0B In Conf Chosen																																																																																												
	0x0C Release Complete Undefined Reason Chosen																																																																																												
	0x0D Facility Call Deflection																																																																																												
	0x0E Security Denied Chosen																																																																																												
	0x0F Release Complete Called Party Not Registered Chosen																																																																																												
	0x10 Release Complete Caller Not Registered Chosen																																																																																												
MSB	0x03 Release Reason Others																																																																																												
LSB	0x01 Outsize Error																																																																																												
	0x02 Protocol Error																																																																																												
	0x03 Socket Open Fail																																																																																												
	0x04 Gatekeeper not Configured																																																																																												
	0x05 ARQ Fail																																																																																												
	0x06 Purge																																																																																												
	0x07 Local End Release																																																																																												
	 MSB 0x04 Q931 Cause Location USER																																																																																												
	LSB Q931 Cause code																																																																																												
	 MSB 0x05 Q931 Cause Location Private Network serving Local user																																																																																												
	LSB Q931 Cause code																																																																																												
	 MSB 0x06 Q931 Cause Location Public Network serving Local user																																																																																												
	LSB Q931 Cause code																																																																																												
	 MSB 0x07 Q931 Cause Location Transit Network																																																																																												
	LSB Q931 Cause code																																																																																												

0x27E4 H.245 Outbound Tunneling

Used in:

Route Control message

Outseize Control message

You can send this TLV in the messages above for each H.323 outbound call initialization (on a per call basis) to configure H.245 tunneling for outbound calls.

This TLV has no effect unless the H.225 Tunneling Configure TLV (0x02D5) is enabled for inbound calls. The IP Signaling card can only use H.245 tunneling if inbound tunneling is enabled.

Byte	Description
0, 1	Tag 0x27E4
2, 3	Length 0x0001
4-7	Value[0-3] 0 - Disabled (default) 1 - Enabled

0x290E NPDI SIP Proxy IP Address

Use this TLV to specify the IP address of the SIP proxy server to which the request is sent. In an outbound call, this TLV specifies where the SIP Messages are sent. The information goes in the request URI of the SIP message.

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

Byte	Description
0, 1	Tag 0x290E
2, 3	Length Variable
:	Value A Null-terminated ASCII string

0x290F NPDI SIP Proxy Port

Use this TLV to specify the port number of the SIP proxy server to which the request is sent. In an outbound call, this TLV specifies where the SIP Messages are sent. The information goes in the request URI of the SIP message.

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

Byte	Description
0, 1	Tag 0x290F
2, 3	Length 0x0004
4-7	Value [0] Variable If this TLV is not present, the configured local port is used in outbound calls (<i>Route Control</i> message) and port 5060 is used for incoming calls for registration (Request for Service and PPL Indication).

0x2910 NPDI SIP Route Type

Use this TLV to specify whether or not a SIP message should be sent to the default proxy. The TLV is used only when the host does not specify the proxy address in the *Route Control* message and the default proxy is desired for a particular call. Absence of this TLV implies that the destination can be reached directly.

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

Byte	Description
0, 1	Tag 0x2910
2, 3	Length 0x01
:	Value 0x00 Direct Route Destination can be reached directly. Do not use default proxy. 0x01 Indirect Route Destination cannot be reached directly. Use default proxy if one is configured.

0x2914 NPDI SIP Local Lookup

Use this TLV to enable or disable local registration lookup for an individual call.

Used in:
 0x0033 NPDI Universal ICB in:
Route Control message

Byte	Description
0, 1	Tag 0x2914
2, 3	Length 0x0001
4	Value 0x00 Disabled 0x01 Enabled

0x2915 NPDI SIP Response Code

This TLV provides the response code received for any SIP message generated by the CSP.

Used in:
 0x0033 NPDI Universal ICB in:
Channel Released with Data message
Release Channel with Data message
PPL Event Indication message
Request for Service with Data message

Byte	Description
0, 1	Tag 0x2915
2, 3	Length 0x0002
4, 5	Value SIP Response Code (2 Bytes) This TLV, when reported in the PPL Event Indication for Event (0x0027/0x0028), will have any valid SIP Response value. The response code to be interpreted for event 0x0027 is as follows: 202 - Accepted (The REFER request is successful) 405 – Method Not Allowed (REFER request failed) 408 – Request Timed Out (REFER request failed) The response code to be interpreted for event 0x0028 is as follows: 100 – Trying 200 - OK (The reference is successful) 503 - Service Unavailable (The reference failed) 603 - Decline (The reference failed)

0x2916 SIP A Leg Transport Type

Reports to the host the default transport type of the A leg portion of the call.

Used in:
 0x0033 NPDI Universal ICB in:
Request for Service with Data message

Byte	Description
0, 1	Tag 0x2916
2, 3	Length 0x0001
:	Value 1 UDP 2 TCP

0x2917 SIP B Leg Transport Type

Reports or sets the default transport type on a per call basis. You set up the default value in the *VoIP Protocol Configure* message with the SIP Transport Type TLV.

Used in:
 0x0033 NPDI Universal ICB in:
Route Control message

Byte	Description
0, 1	Tag 0x2917
2, 3	Length 0x0001
:	Value 1 UDP 2 TCP

0x2919 NPDI SIP To Username

In a *Request for Service* message and PPL Event Indication, this TLV provides information parsed from the “To” header field of the incoming SIP message (for example INVITE or REGISTER). This TLV could be used in a *Route Control* message to specify the remote User Agent’s addressing information, which would be used to create the “To” header for the outgoing SIP message. Use this TLV to specify a username in the “To” header.

Used in:

0x0033 NPDI Universal ICB in:

Outseize Control message

Request for Service with Data message

Route Control message

Byte	Description
0, 1	Tag 0x2919
2, 3	Length Variable
:	Value A Null-terminated ASCII string

0x291A NPDI SIP To Password

In a *Request for Service* message and PPL Event Indication, this TLV provides information parsed from the “To” header field of the incoming SIP message (for example INVITE or REGISTER). This TLV could be used in a *Route Control* message to specify the remote User Agent’s addressing information, which would be used to create the “To” header for the outgoing SIP message.

Use this TLV to specify a password in the “To” header.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Route Control message

Byte	Description
0, 1	Tag 0x291A
2, 3	Length Variable
:	Value A Null-terminated ASCII string

0x291B NPDI SIP To Host Name

In a *Request for Service* message and PPL Event Indication, this TLV provides information parsed from the “To” header field of the incoming SIP message (for example INVITE or REGISTER). This TLV could be used in a *Route Control* message to specify the remote User Agent’s addressing information, which would be used to create the “To” header for the outgoing SIP message.

Use this TLV to specify an IP address in the “To” header

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Route Control message

Byte	Description
0, 1	Tag 0x291B
2, 3	Length Variable
:	Value A Null-terminated ASCII string

0x291C NPDI SIP To Port

In a *Request for Service* message and PPL Event Indication, this TLV provides information parsed from the “To” header field of the incoming SIP message (for example INVITE or REGISTER). This TLV could be used in a *Route Control* message to specify the remote User Agent’s addressing information, which would be used to create the “To” header for the outgoing SIP message.

This TLV provides the port from the “To” address URL.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Route Control message

Byte	Description
0, 1	Tag 0x291C
2, 3	Length 0x0004
4-7	Value Variable If this TLV is not present, a default value of 5060 is used.

0x291E NPDI SIP Refer to Username

Use this TLV in a *PPL Event Request* message that is used to generate a REFER message. This TLV specifies the referred target User Agent’s addressing information which is used to create the Refer-To header for the outbound REFER request. Use this TLV to specify a username in the Refer-To header.

Used in:

0x0033 NPDI Universal ICB in:

PPL Event Request message

Byte	Description
0, 1	Tag 0x291E
2, 3	Length Variable
:	Value[0] A NULL-terminated ASCII string

0x291F NPDI SIP Refer to Password

Use this TLV in a *PPL Event Request* message that is used to generate a REFER message. This TLV specifies the referred target User Agent's addressing information which is used to create the Refer-To header for the outbound REFER request. Use this TLV to specify a password in the Refer-To header.

Used in:

0x0033 NPDI Universal ICB in:

PPL Event Request message

Byte	Description
0, 1	Tag 0x291F
2, 3	Length Variable
:	Value[0] A NULL-terminated ASCII string

0x2920 NPDI SIP Refer to Host Name

Use this TLV in a *PPL Event Request* message that is used to generate a REFER message. This TLV specifies the referred target User Agent's addressing information which is used to create the Refer-To header for the outbound REFER request. Use this TLV to specify a host IP address in the Refer-To header.

Used in:

0x0033 NPDI Universal ICB in:

PPL Event Request message

Byte	Description
0, 1	Tag 0x2920
2, 3	Length Variable
:	Value[0] A NULL-terminated ASCII string

0x2921 NPDI SIP Refer to Port

Use this TLV in a *PPL Event Request* message that is used to generate a REFER message. This TLV specifies the referred target User Agent's addressing information which is used to create the Refer-To header for the outbound REFER request. Use this TLV to specify the port in the Refer-To header.

Used in:

0x0033 NPDI Universal ICB in:

PPL Event Request message

Byte	Description
0, 1	Tag 0x2921
2, 3	Length 0x0004
:	Value[0] If this TLV is not present, the default 5060 is used.

0x2922 SIP REFER-To Header Param

This TLV contains the Refer-To Parameter in the Refer-To Header field:

Used in:

PPL Event Request message

PPL Event Indication message

Byte	Description
0, 1	Tag 0x2922
2,3	Length Variable (Maximum 250 bytes)
4-n	Value[0] A null-terminated ASCII string

- The data in the TLV 0x2922 is up to the interpretation of the host. The SIP stack does no masking/unmasking.
- If the TLV 0x2922 is present in the PPL ER for REFER then the host should have provided the '>' in the TLV data itself. If the host does not provide it then it is assumed to be at the end by the SIP stack.

0x2923 NPDI SIP From User Name

In a *Request for Service* message and PPL Event Indication, this TLV provides information parsed from the "From" header field of the incoming SIP message (for example INVITE or REGISTER). This

TLV could be used in a *Route Control* message to specify the local User Agent's addressing information, which would be used to create the "From" header for the outgoing SIP message.

Use this TLV to specify a user name in the "From" header.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Route Control message

Byte	Description
0, 1	Tag 0x2923
2, 3	Length Variable
:	Value A Null-terminated ASCII string

0x2924 NPDI SIP From Password

In an *Request for Service* message and PPL Event Indication, this TLV provides information parsed from the "From" header field of the incoming SIP message (for example INVITE or REGISTER). This TLV could be used in a *Route Control* message to specify the local User Agent's addressing information, which would be used to create the "From" header for the outgoing SIP message.

Use this TLV to specify a password in the "From" header.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Route Control message

Byte	Description
0, 1	Tag 0x2924
2, 3	Length Variable
:	Value A Null-terminated ASCII string

0x2925 NPDI SIP From Host Name

In an *Request for Service* message and PPL Event Indication, this TLV provides information parsed from the "From" header field of the incoming SIP message (for example INVITE or REGISTER). This TLV could be used in a *Route Control* message to specify the local User Agent's addressing information, which would be used to create the "From" header for the outgoing SIP message.

Use this TLV to specify an IP address in the “From” header.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Request for Service message

Byte	Description
0, 1	Tag 0x2925
2, 3	Length Variable
:	Value A Null-terminated ASCII string

0x2926 NPDI SIP From Port

In a *Request for Service* message and PPL Event Indication, this TLV provides information parsed from the “From” header field of the incoming SIP message (for example INVITE or REGISTER). This TLV could be used in a *Route Control* message to specify the local User Agent’s addressing information, which would be used to create the “From” header for the outgoing SIP message.

This TLV provides the port from the “From” URL.

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

Request for Service with Data message

Byte	Description
0, 1	Tag 0x2926
2, 3	Length 0x0004
4-7	Value Variable If this TLV is not present, the configured local port is used in outbound calls (Route Control message) and port 5060 is used for incoming calls for registration (Request for Service and PPL Indication).

0x2928 SIP From Display Name

This TLV contains the user name parsed from the display name in ASCII format.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Route Control message

Outseize Control message

Byte	Description
0, 1	Tag 0x2928
2, 3	Length Variable (Maximum of 250 bytes)
4-n	Value A Null-terminated ASCII string containing From Display Name

0x2929 - SIP Referred By Header URI TLV

This TLV does the following:

- Instructs the CSP SIP stack to insert Referred-By header in the outbound REFER. In case no data is provided CSP SIP stack fills in the contact URI.
- Reports the URI in the Referred-By header, if present, in the received REFER or INVITE request.
- Writes the URI of the Referred-By header in the outbound INVITE request.

Used in:

Route Control

Outseize Control

Request For Service with Data

PPL Event Request

PPL Event Indication

Byte	Description
0, 1	Tag 0x2929
2, 3	Length Variable (Maximum is 100)
4-n	Value[0-3] Null Terminated ASCII string

0x292A - SIP Referred By Header Parameters TLV

This TLV does the following:

- Reports the Referred-By header parameters, if present, in the received REFER or INVITE request.
- Writes the parameters of the Referred-By header in the outbound REFER or INVITE request.

Used in:

Route Control
Outseize Control
Request For Service with Data
PPL Event Request
PPL Event Indication

Byte	Description	
0, 1	Tag	0x292A
2, 3	Length	Variable (Maximum is 100)
4-n	Value	Null Terminated ASCII string

0x292D NPDI SIP Contact URI User Name

This TLV contains the user name parsed from the contact header in ASCII format.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Byte	Description	
0, 1	Tag	0x292D
2, 3	Length	Variable
4-n	Value	NULL Terminated ASCII string for SIP Contact user name

0x292E NPDI SIP Contact Password

This TLV contains the password in the contact header in ASCII format.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Byte	Description
0, 1	Tag 0x292E
2, 3	Length Variable
	Value NULL Terminated ASCII value for SIP Contact password

0x292F NPDI SIP Contact Hostname

This TLV provides the IP address from the Contact URL in ASCII string.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Byte	Description
0, 1	Tag 0x292F
2, 3	Length Variable
	Value NULL Terminated ASCII value for SIP 'Contact' hostname

0x2930 NPDI SIP Contact Port

This TLV provides the port from Contact URL.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Byte	Description
0, 1	Tag 0x2930
2, 3	Length 0x0004
4-7	Value Variable If this TLV is not present, a default value of 5060 is used.

0x2931 NPDI SIP Contact Expires

This TLV defines the seconds remaining before the Contact expires.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Byte	Description
0, 1	Tag 0x2931
2, 3	Length 0x0004
4-7	Value [0-3] Variable

0x2933 SIP Contact Transport Type

Specifies the transport type contained in the registration method.

Used in:

0x0033 NPDI Universal ICB in:

Setup Indication message

PPL Event Indication message

Byte	Description
0, 1	Tag 0x2933
2, 3	Length 0x0001
4-7	Value[0] 1 - UDP 2 - TCP

0x2934 SIP Contact URI User Information Qualifier

Use this TLV to append a user-defined ASCII string to the Contact URI user name in outbound SIP calls from the CSP. Specifically, the ASCII string is populated in the user information of the Contact-URI header in the outbound SIP INVITE message.

When the CSP is in a SIP/PSTN gateway environment, you can use this TLV to report the originating trunk group information in a PSTN-to-SIP call.

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

Outseize Control message

Byte	Description
0, 1	Tag 0x2934
2, 3	Length Variable
4-n	Value[0-n] Null-terminated ASCII string

0x2935 SIP Contact URI Parameters

Use this TLV to receive and send Contact-URI parameters to and from host applications.

For inbound calls, this TLV carries all of the Contact URI parameters received by the SIP software in the *Request for Service with Data* message to host applications.

For outbound SIP calls, this TLV allows you to insert Contact URI parameters in SIP INVITE messages.

To report this TLV you must first set Bit 5 in the TLV *0x027F SIP Message Information Mask (4-86)*.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Route Control message

Outseize Control message

Byte	Description
0, 1	Tag 0x2935
2, 3	Length Variable
4-n	Value[0-n] Null-terminated ASCII string for URI parameters in the CONTACT-URI.

0x2936 SIP Tunnel Type

For all outgoing messages, this TLV enables and disables the use of tunneled data and specifies the tunneled data type.Used in:

0x0033 NPDI Universal ICB in:

Route Control message
Outseize Control message
PPL Event Request message

Byte	Description
0, 1	Tag 0x2936
2, 3	Length 0x0002
4-6	Value[0-2] 0x0001 No tunneling (Default) 0x0002 CSP UPDF Tunneling 0x0004 Custom MIME Body

0x2937 NPDI SIP Authenticate Scheme

Use this TLV to specify the authentication scheme: basic or digest (a scheme based on cryptographic hashes).

Used in:

0x0033 NPDI Universal ICB in:

PPL Event Request message with Event IDs 0x0C

Byte	Description
0, 1	Tag 0x2937
2, 3	Length 0x0001
4, 5	Value[0-1] 0x00 Basic 0x01 Digest

0x2938 NPDI SIP Authentication Realm

The TLV specifies the realm used in the www-authenticate header.

Used in:

0x0033 NPDI Universal ICB in:

PPL Event Request message with Event IDs 0x0C

Byte	Description
0, 1	Tag 0x2938
2, 3	Length Variable
4-n	Value[0-n] Null-terminated ASCII string for realm

0x2939 NPDI SIP Authenticate Username

Use this TLV to specify the username used to authenticate the request.

Used in:

0x0033 NPDI Universal ICB in:

PPL Event Request message with Event IDs 0x0C

Byte	Description
0, 1	Tag 0x2939
2, 3	Length Variable
4-n	Value[0-n] NULL terminated ASCII string for username

0x293A NPDI SIP Authenticate Password

Use this TLV to specify the password used to authenticate the request.

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

PPL Event Request message with Event IDs 0x0C

Byte	Description
0, 1	Tag 0x293A
2, 3	Length Variable
4-n	Value[0-n] NULL Terminated ASCII string for password

0x293B NPDI SIP Authorization Username

Use this TLV to specify the username used to generate the Authorization header for an outgoing request when the CSP receives a 401 response.

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

Byte	Description
0, 1	Tag 0x293B
2, 3	Length Variable
4-n	Value[0-n] NULL Terminated ASCII string for username

0x293C NPDI SIP Authorization Password

Use this TLV to specify the password used to generate the Authorization header for an outgoing request when the CSP receives a 401 response.

Used in:
 0x0033 NPDI Universal ICB in:
Route Control message

Byte	Description
0, 1	Tag 0x293C
2, 3	Length Variable
4-n	Value[0-n] NULL Terminated ASCII string for password

0x293D NPDI SIP Proxy Authorization Username

Use this TLV to specify the username used to generate Proxy-Authorization header for an outgoing request when the CSP receives a 407 response.

Used in:
 0x0033 NPDI Universal ICB in:
Route Control message

Byte	Description
0, 1	Tag 0x293D
2, 3	Length Variable
4-n	Value[0-n] NULL Terminated ASCII string for username

0x293E NPDI SIP Proxy Authorization Password

Use this TLV to specify the password used to generate the Proxy-Authorization header for an outgoing request when the CSP receives a 407 response.

Used in:
 0x0033 NPDI Universal ICB in:
Route Control message

Byte	Description
0, 1	Tag 0x293E
2, 3	Length Variable
4-n	Value[0-n] NULL Terminated ASCII value for password

0x293F NPDI SIP Authentication Timeout

Use this TLV to specify the timeout for authentication response.

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

PPL Event Request message with Event IDs 0x0C

Byte	Description
0, 1	Tag 0x293F
2, 3	Length 0x01
4-n	Value[0-n] Timeout in seconds Minimum: 30 seconds Maximum: 255 seconds

0x2941 SIP Authorization Header

Use this TLV to specify the authorization header value in a SIP message.

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

Request for Service with Data message

Byte	Description
0, 1	Tag 0x2941
2, 3	Length Variable
4-n	Value[0-n] Null Terminated ASCII value for Authorization header.

0x2942 SIP Proxy Authorization Header

Use this TLV to specify the SIP Proxy authorization header value in a SIP message.

Used in:

0x0033 NPDI Universal ICB in:

Route Control message

Request for Service with Data message

Byte	Description
0, 1	Tag 0x2942
2, 3	Length Variable
4-n	Value[0-n] Null Terminated ASCII value for Proxy-Authorization header.

0x2943 SIP Authenticate Header

Use this TLV to specify the WWW-Authenticate header value in a SIP message.

Used in:

0x0033 NPDI Universal ICB in:

Release Channel with Data message

Byte	Description
0, 1	Tag 0x2943
2, 3	Length Variable
4-n	Value[0-n] Null Terminated ASCII value for WWW-Authzication header.

0x2944 SIP Proxy-Authenticate Header

Use this TLV to specify the Proxy-Authenticate header value in a SIP message.

Used in:

0x0033 NPDI Universal ICB in:

Release Channel with Data message

Byte	Description
0, 1	Tag 0x2944
2, 3	Length Variable
4-n	Value[0-n] Null Terminated ASCII value for Proxy-Authzication header.

0x2946 - SIP Request URI Password

Used in:

PPL Event Request message

PPL Event Indication message

Byte	Description
0, 1	Tag 0x2946
2, 3	Length Variable (maximum 20 bytes)
4-n	Value[0-n] Null-terminated ASCII string

0x2947 - SIP Request URI Header

Used in:

PPL Event Request message

PPL Event Indication message

Byte	Description
0, 1	Tag 0x2947
2, 3	Length Variable (maximum of 650 bytes)
4-n	Value[0-n] Null-terminated ASCII string

0x2949 - SIP Response Reason Phrase

This TLV is used when reporting or generating the 182 Reason Phrase.

Not all characters are allowed in the Reason Phrase of a SIP response. Refer to the SIP RFC ABNF for a complete list.

Byte	Description
0, 1	Tag 0x2949
2, 3	Length Variable (Maximum of 250)
4-n	Value Null terminated ASCII string

0x294A - NPDI SIP Extensions

Used in:

Request for Service

PPL Event Indication message

Byte	Description
0, 1	Tag 0x294A
2,3	Length 0x0004
4-7	<p>Value This field is a 32-bit mask. Each bit selects specific SIP Extension SIP message fields and is listed from LSB to MSB.</p> <p>0 - Disable 1 - Enable</p> <p>Supported header tags Bit 0 - Timer Bit 1 - 100rel (PRACK) Bits 2-15 - Reserved</p> <p>Required header tags Bit 16 - Timer Bit 17 - 100rel(PRACK) Bits 18-31 - Reserved</p>

0x294B Notify Status

Used in:

0x0033 NPDI Universal ICB in:

Byte	Description
0, 1	Tag 0x294B
2,3	Length 0x0002
4, 5	Value (0,1) Any appropriate SIP Response value. (Typically 200 OK or 600 Decline)

0x294C Refer Spans Channel

Used in:

0x0033 NPDI Universal ICB in:

Byte	Description
0, 1	Tag 0x294C
2,3	Length 0x0003
4,5	Value(0,1) Span Number
6	Value(2) Channel Number

0x294E SIP Remote IP Address

This TLV reports the IP Address of the remote socket.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service message

Byte	Description
0, 1	Tag 0x294E
2, 3	Length 0x0004
4-n	Value[0-n] MSB of the IP Address LSB of the IP Address

0x294F SIP Remote IP Port

This TLV reports the IP Port of the remote socket.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service message

Byte	Description
0, 1	Tag 0x294F
2, 3	Length 0x0002
4-n	Value[0-n] MSB of the IP Port LSB of the IP Port

0x2950 SIP Call ID

Use this TLV to specify the Call ID of the SIP message.

Used in:

Route Control message (ACK only)

Outseize Control message (ACK only)

Request for Service with Data message

Byte	Description
0, 1	Tag 0x2950
2, 3	Length Variable
4-n	Value[0-n] Null Terminated ASCII value of Call-ID header.

0x2951 SIP From Tag

Use this TLV to specify the Tag value in the From Header of the SIP message.

Used in:

Request for Service with Data message

Byte	Description
0, 1	Tag 0x2951
2, 3	Length Variable
4-n	Value[0-n] Null Terminated ASCII value of Call-ID header.

0x2952 SIP to Tag

Use this TLV to specify the Tag value in the To Header of the SIP message.

Used in:

Request for Service with Data message

Byte	Description
0, 1	Tag 0x2952
2, 3	Length Variable
4-n	Value[0-n] Null Terminated ASCII value of Tag in To header

0x2953 Subsequent Information Status

If all the NPDI data cannot be sent from the CSP to the host in one *Request for Service (RFS) with Data* message, the remaining data is sent in the *PPL Event Indication* message (component ID 0x00A7 and event ID 0x23). If the CSP sends the data in multiple messages, the *RFS with Data* and subsequent PPL event indications, the CSP adds this TLV to those messages. This TLV indicates the sequence number of the message and whether there is more data to follow in the PPL Event Indication.

Similarly, if all the NPDI data cannot be sent from the host in the *Route Control* or *Outseize Control* message, the remaining data is sent in one or more *PPL Event Request* messages. Each of these messages has to include the proper sequence number.

Used in:

Outseize Control message

Route Control message

PPL Event Request message

Request for Service with Data message

PPL Event Indications message

Byte	Description
0, 1	Tag 0x2953
2, 3	Length 0x0002
4	Value[0] Sequence number of the message The <i>Request for Service</i> message has a value of 1. Subsequent PPL Event Indications have a value of 1+n where n is the number of the PPL Event Indication. The <i>Route Control</i> or <i>Outseize Control</i> message have a value of 1. Subsequent PPL Event Requests have a value of 1+n where n is the number of the <i>PPL Event Request</i> message.
5	Value[1] Last message indicator 0 there are more messages to follow 1 the current message is the last one

0x2954 SIP Request URI User Name

Used to specify the user name in the Request URI. If the host provides the Request URI parameters, it must also provide the “To Header” parameters (see TLVs 0x2919, 0x291A, 0x291B, 0x291C). The host must also provide the URI host name and port name (See TLVs 0x2955 and 0x2956).

Used in:

0x0033 NPDJ Universal ICB in:

Route Control message

Request for Service with Data message

Byte	Description
0, 1	Tag 0x2954
2, 3	Length Variable
4-n	Value[0-n] Null Terminated ASCII for user name in Request URI.

0x2955 SIP Request URI Host Name

Used to specify the host name in the Request URI. If the host provides the Request URI parameters, it must also provide the “To Header” parameters. (See TLVs 0x2919, 0x291A, 0x291B, 0x291C). The host must also provide the URI user name and port name (See TLVs 0x2954 and 0x2956).

Used in:

0x0033 NPDJ Universal ICB in:

Route Control message

Request for Service with Data message

Byte	Description
0, 1	Tag 0x2955
2, 3	Length Variable
4-n	Value[0-n] Null Terminated ASCII for host name in Request URI.

0x2956 SIP Request URI Port

Used to specify the port name in the Request URI. If the host provides the Request URI parameters, it must also provide the “To Header” parameters. (See TLVs 0x2919, 0x291A, 0x291B, and 0x291C). The host must also provide the URI host name and user name (See TLVs 0x2954 and 0x2956).

Used in:

*0x0033 NPD Universal ICB in:
Route Control message*

Byte	Description
0, 1	Tag 0x2956
2, 3	Length 0x0004
4	Data[0] MSB Port in Request URI Data[1] : Data[2] : Data[3] LSB Port in Request URI

0x2957 SIP Request URI User Information Qualifier

Use this TLV to append a user-defined ASCII string to the Request URI user name in outbound SIP calls from the CSP. Specifically, the ASCII string is populated in the user information of the Request-URI header in the outbound SIP INVITE message.

When the CSP is in a SIP/PSTN gateway environment, you can use this TLV to report the terminating trunk group information in a PSTN-to-SIP call.

If the host provides the Request URI parameters, it must also provide the “To Header” parameters. (See TLVs 0x2919, 0x291A, 0x291B, and 0x291C).

Used in:

*0x0033 NPD Universal ICB in:
Route Control message
Outseize Control message*

Byte	Description
0, 1	Tag 0x2957
2, 3	Length Variable
4-7	Value[0-n] Null-terminated ASCII string

0x2958 SIP Request URI Parameters

Use this TLV to receive and send proprietary Request URI Parameters to and from host application. For inbound calls, this TLV carries all of the Request URI parameters received by the SIP stack in the *Request for Service with Data* message to host applications.

To report this TLV to the host, you must program the new Bit 6 in the SIP Message Information Mask TLV (0x027F) used in the *VoIP Protocol Configure* message (0x00EE).

For outbound calls, host application developers can use this TLV in the *Route Control* and *Outseize Control* messages to insert Request URI parameters in SIP INVITE messages.

Used in:

0x0033 NPDI Universal ICB in:

Request for Service with Data message

Route Control message

Outseize Control message

Byte	Description
0, 1	Tag 0x2958
2, 3	Length Variable
4-n	Value[0-n] Null-terminated ASCII string

0x2959 SIP Remote Party ID

Used in:

Request for Service with Data (0x002D)

PPL Event Indication (0x0043)

PPL Event Request (0x0044)

Route Control (0x00E8)

Outseize Control (0x002C)

Byte	Description
0, 1	Tag 0x2959
2, 3	Length Variable (Maximum of 250 bytes when used in Outseize or Route Control or subsequent PPL Event Request messages)
4-n	Value Contains the value of the Remote Party ID header as a null-terminated ASCII string.

0x295A SIP RPID Privacy

Used in:

Request for Service with Data (0x002D)

PPL Event Indication (0x0043)

PPL Event Request (0x0044)

Route Control (0x00E8)

Outseize Control (0x002C)

Byte	Description	
0, 1	Tag	0x295A
2, 3	Length	Variable (Maximum of 250 bytes when used in Outseize or Route Control or subsequent PPL Event Request messages)
4-n	Value	Contains the value of the RPID Privacy header as a null terminated ASCII string.

0x295B SIP Subject Header

The subject header is also supported in the INFO message. The CSP allows the host for read and write access to the Subject header. The Subject Header TLV (0x295B) reports the Subject header within the PPL Event Indication for an inbound INFO message. This reporting of the Subject header in the INFO message is not configurable.

The Subject Header TLV (0x295B) is used in the PPL Event Request for outbound INFO to include the subject header. This TLV is also used in the PPL Event Request message for an outbound INFO message to include the subject header.

PPL Event Request message

PPL Event Indication message

Byte	Description	
0, 1	Tag	0x002B
2, 3	Length	Variable (Maximum of 250)
4-n	Value	Null terminated ASCII string

0x295C SIP Message Body

This TLV adds a message body to a SIP message. It also reports the message body (other than Content Type) if present, in a SIP message. Prior to this feature, this TLV added/reported the message body only for the SIP INFO message. The SIP stack adds a new line after the end

of the message body. The host does not need to include this line in the TLV. The new line at the end of the message body will not be reported in this TLV.

PPL Event Request message

PPL Event Indication message

Byte	Description	
0, 1	Tag	0x002C
2, 3	Length	Variable (Maximum of 500)
4-n	Value	Null terminated ASCII string

0x295D Content Type

This TLV adds the content type to a SIP message. It also reports the content type, if present, in a SIP message. Prior to this feature, this TLV adds/reports content type only for the SIP INFO message. If there is a valid message body but the host has not supplied the content type, then the SIP stack adds the following content type as the default: “Content-Type: application/custom”

PPL Event Request message

PPL Event Indication message

Byte	Description	
0, 1	Tag	0x002D
2, 3	Length	Variable (Maximum of 100)
4-n	Value	Null terminated ASCII string

0x295E SIP Hostname

This TLV contains the Hostname in the Header Field.

Byte	Description
0, 1	Tag 0x295E
2, 3	Length Variable
4-n	Value A null-terminated ASCII string

0x295F SIP Port Number

This TLV contains the Port Number in the Header Field.

Byte	Description
0, 1	Tag 0x295F
2, 3	Length 0x0004
4-7	Value Variable

0x2960 Privacy Header

This TLV reports the “Privacy” header if present in the inbound INVITE message. The following are the defined set of Privacy header contents but they are not limited to these.

“header” / “session” / “user” / “none” / “critical” / “id”

Byte	Description
0, 1	Tag 0x2960
2, 3	Length Variable (Maximum of 250)
4-n	Value Null terminated ASCII string

0x2961 Private Header Container

This TLV contains nested TLVs that in turn contain the P-Header type and P-Header content. This TLV uses the following nested TLVs:

- 0x2962 Private Header Type TLV
- 0x2963 Private Header Data TLV

This TLV along with the two nested TLVs is reported in the RFS as many number of times as the number of P-Headers appear in the INVITE message. The total length of the NPDI data that can be reported to the host limits this number.

Byte	Description
0, 1	Tag 0x2961
2, 3	Length Variable (Maximum of 250)

0x2962 Private Header Type

This TLV contains the P-header type. Use it within the Private Header Container TLV (0x2961) as a nested TLV and in combination with the Private Header data TLV (0x2963). It does not have any meaning if used as a standalone TLV.

Byte	Description
0, 1	Tag 0x2962
2, 3	Length 0x0001
4-n	Value 0x01 - P-Asserted-Identity 0x02 - P-Preferred-Identity 0x03 - P-Access-Network-Info

0x2963 Private Header Data

This TLV contains the content of the P-header. Use it within the Private Header info TLV (0x2961) as a nested TL and in combination with the Private Header Type TLV (0x2962). It does not have any meaning if used as a standalone TLV.

Byte	Description
0, 1	Tag 0x2963
2, 3	Length Variable (Maximum of 241)
4-n	Value Null terminated ASCII string

0x2964 SIP MIME Information

Used in:

0x0033 NPDI Universal ICB in:

Route Control message*Outsize Control* message*Release Channel with Data**Connect with Data**Channel Released with Data**Request for Service with Data* message*PPL Event Request* message*PPL Event Indication* message

Byte	Description
0, 1	Tag 0x2964
2, 3	Length Variable (Maximum of 780 bytes)

0x2965 SIP Content Encoding

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2965
2, 3	Length Variable
4-n	Value[0-n] Null Terminated ASCII String

0x2966 SIP Content Disposition

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2966
2, 3	Length Variable
4-n	Value[0-n] Null Terminated ASCII String

0x2967 SIP Content Language

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2967
2, 3	Length Variable
4-n	Value[0-n] Null Terminated ASCII String

0x2968 SIP MIME Message Body

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2968
2, 3	Length Variable
4-n	Value[0-n] Null Terminated ASCII String

0x2969 SIP Session Expiry Interval

This TLV reports the session expiry interval used in the Re-transmit INVITE message.

Byte	Description
0, 1	Tag 0x2969
2, 3	Length 0x0004
4-8	Value Timer Value

0x2970 SIP Minimum Session Expiry Interval

This TLV reports the minimum session expiry interval used in the Re-transmit INVITE message .

Byte	Description
0, 1	Tag 0x2970
2, 3	Length 0x0004
4-8	Value Timer Value

0x299A SIP Header Field Container

For SIP calls, this TLV contains the nested TLVs that in turn contain information regarding SIP Header Fields.

Used in:
0x0033 NPDI Universal ICB

This TLV uses the following nested TLVs:

- 0x299B SIP Header Field ID
- 0x299C SIP Header Field

Byte	Description
0, 1	Tag 0x299A
2, 3	Length Variable

0x299B SIP Header Field ID

This TLV contains the ID of the Header Field being reported.

Used in:
0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x299B
2,3	Length 0x0002
4,5	Value[0] Valid value: 0x0022 - Via

0x299C SIP Header Field

For SIP calls, this TLV contains the nested TLVs that in turn contain the SIP Header Field Parameters.

Used in:
0x0033 NPDI Universal ICB

This TLV uses the following nested TLVs:

- 0x295E SIP Hostname
- 0x295F SIP Port Number

Byte	Description
0, 1	Tag 0x299C
2, 3	Length Variable

0x29D4 SIP Do Not Close Socket

Allows an out bound call via TCP to keep its socket. Normally TCP sockets are closed after a period of time after the last message in a transaction is sent. If you include this TLV in a *Route Control* message that causes the use of a TCP socket, the socket will not be closed until explicitly commanded to close by the host with a PPL Event Request.

Used in:

*0x0033 NPDI Universal ICB in:
Route Control* message

Byte	Description
0, 1	Tag 0x29D4
2, 3	Length 0x0001
4	Value[0] 0x00 Close socket as if this TLV were not sent 0x01 Do not close this socket (persistent mode)

0x29FF Media Local End Point Information

SIP

For SIP calls, this TLV contains the nested TLVs that in turn contain the entire Session Description Protocol (SDP) in NPDI format. In the *Request for Service* or *PPL Event Indication* messages, this TLV cannot exceed 250 bytes.

Used in: 0x0033 NPDI Universal ICB

RTP

This TLV uses the following nested TLVs that contain the Local End-Point Media Information.

- 0x2A07 Media Port
- 0x2A0E Media Connection Address

Byte	Description
0, 1	Tag 0x29FF
2, 3	Length Variable

0x2A00 Media Remote End Point Information

SIP

For SIP calls, this TLV contains the nested TLVs that in turn contain the entire Session Description Protocol (SDP) in NPDI format.

Used in: 0x0033 NPDI Universal ICB

RTP

This TLV uses the following nested TLVs that contain the Remote End-Point Media Information.

- 0x2A07 Media Port
- 0x2A0E Media Connection Address

Byte	Description
0, 1	Tag 0x2A00
2, 3	Length Variable

0x2A01 Media Per Stream Information

SIP

For SIP calls, this TLV contains the nested TLVs that in turn contain the media stream in NPDI format.

Used in: 0x0033 NPDI Universal ICB

RTP

For Media Streaming over RTP, this TLV is a nested TLV that contains the media stream information. This TLV is nested in the following TLVs in the *Play File Start 0x011B* message:

0x29FF Media Local End Point Information

0x2A00 Media Remote End Point Information

Byte	Description
0, 1	Tag 0x2A01
2, 3	Length Variable

0x2A02 Media Per Codec Information

SIP

This TLV contains the nested TLVs that in turn contain information about each codec per media stream.

Used in: 0x0033 NPDI Universal ICB

RTP

Byte	Description
0, 1	Tag 0x2A02
2, 3	Length Variable

0x2A03 Media Type

SIP

This TLV defines the type of media for each media stream.

Used in: 0x0033 NPDI Universal ICB

RTP

Byte	Description
0, 1	Tag 0x2A03
2, 3	Length 0x0001
4	Value Media Type 0xFF Invalid 0x00 Audio 0x01 Video (Requires Call Agent.)

0x2A07 Media Port

This TLV defines the port for each media stream.

SIP

Used in: 0x0033 NPDI Universal ICB

RTP

Byte	Description
0, 1	Tag 0x2A07
2, 3	Length 0x0004
4-7	Value [0-3] Media Port

0x2A08 Media Payload Type

This TLV carries the actual payload type ID from either the NPDI static payload types or from network signaling data for dynamic payload types.

SIP

Used in: 0x0033 NPDI Universal ICB

RTP

Byte	Description																																												
0, 1	Tag 0x2A08																																												
2, 3	Length 0x0002																																												
4	Value(0) Payload ID 00 Static 01 Dynamic																																												
5	Value(1) Codec Type Numeric codec value based on NPDI codec type (decimal) <table border="0" style="width: 100%;"> <tr> <td style="width: 50%;">1 G.711A 64K</td> <td style="width: 50%;">23 DVI4 8K</td> </tr> <tr> <td>2 G.711U 64K</td> <td>24 DVI4 11K</td> </tr> <tr> <td>3 G.726 16K</td> <td>25 DVI4 16K</td> </tr> <tr> <td>4 G.726 24K</td> <td>26 DVI4 22K</td> </tr> <tr> <td>5 G.726 32K</td> <td>27 LPC</td> </tr> <tr> <td>6 G.726 40K</td> <td>28 G.722 64K</td> </tr> <tr> <td>7 G.727 16K</td> <td>29 L16 2CH</td> </tr> <tr> <td>8 G.727 24K 16K</td> <td>30 L16 1CH</td> </tr> <tr> <td>9 G.727 24K</td> <td>31 MPA</td> </tr> <tr> <td>10 G.727 32K 16K</td> <td>32 CELB</td> </tr> <tr> <td>11 G.727 32K 24K</td> <td>33 JPEG</td> </tr> <tr> <td>12 G.727 32K</td> <td>34 NV</td> </tr> <tr> <td>13 G.727 40K 16K</td> <td>35 H261</td> </tr> <tr> <td>14 G.727 40K 24K</td> <td>36 MPV</td> </tr> <tr> <td>15 G.727 40K 32K</td> <td>37 MP2T</td> </tr> <tr> <td>16 G.723 63K</td> <td>38 H263</td> </tr> <tr> <td>17 G.723 53K</td> <td>39 G.711A 56K</td> </tr> <tr> <td>18 G.729</td> <td>40 G.711U 56K</td> </tr> <tr> <td>19 G.728</td> <td>41 G.722 56K</td> </tr> <tr> <td>20 1016</td> <td>42 G.722 48K</td> </tr> <tr> <td>21 G.721</td> <td>255 Invalid</td> </tr> <tr> <td>22 GSM</td> <td></td> </tr> </table>	1 G.711A 64K	23 DVI4 8K	2 G.711U 64K	24 DVI4 11K	3 G.726 16K	25 DVI4 16K	4 G.726 24K	26 DVI4 22K	5 G.726 32K	27 LPC	6 G.726 40K	28 G.722 64K	7 G.727 16K	29 L16 2CH	8 G.727 24K 16K	30 L16 1CH	9 G.727 24K	31 MPA	10 G.727 32K 16K	32 CELB	11 G.727 32K 24K	33 JPEG	12 G.727 32K	34 NV	13 G.727 40K 16K	35 H261	14 G.727 40K 24K	36 MPV	15 G.727 40K 32K	37 MP2T	16 G.723 63K	38 H263	17 G.723 53K	39 G.711A 56K	18 G.729	40 G.711U 56K	19 G.728	41 G.722 56K	20 1016	42 G.722 48K	21 G.721	255 Invalid	22 GSM	
1 G.711A 64K	23 DVI4 8K																																												
2 G.711U 64K	24 DVI4 11K																																												
3 G.726 16K	25 DVI4 16K																																												
4 G.726 24K	26 DVI4 22K																																												
5 G.726 32K	27 LPC																																												
6 G.726 40K	28 G.722 64K																																												
7 G.727 16K	29 L16 2CH																																												
8 G.727 24K 16K	30 L16 1CH																																												
9 G.727 24K	31 MPA																																												
10 G.727 32K 16K	32 CELB																																												
11 G.727 32K 24K	33 JPEG																																												
12 G.727 32K	34 NV																																												
13 G.727 40K 16K	35 H261																																												
14 G.727 40K 24K	36 MPV																																												
15 G.727 40K 32K	37 MP2T																																												
16 G.723 63K	38 H263																																												
17 G.723 53K	39 G.711A 56K																																												
18 G.729	40 G.711U 56K																																												
19 G.728	41 G.722 56K																																												
20 1016	42 G.722 48K																																												
21 G.721	255 Invalid																																												
22 GSM																																													

0x2A09 Media Payload Description

This TLV carries the numerical mapping of a textual payload type string, received for example in the SDP RTPMAP attribute. This TLV is typically present when the Media Payload Type TLV carries a dynamic payload type number.

SIP

Used in:
0x0033 NPDI Universal ICB:

Byte	Description
0, 1	Tag 0x2A09
2, 3	Length 0x0001
4	Value Payload type string number (decimal)
	1 PCMA
	2 PCMU
	3 G726
	7 G727
	16 G723
	18 G729
	19 G728
	20 1016
	21 G721
	22 GSM
	23 DVI4
	27 LPC
	28 G722
	29 L16
	31 MPA
	32 CelB
	33 JPEG
	34 NV
	35 H261
	36 MPV
	37 MP2T
	38 H263
	50 Telephone event
	51 Red - (Redundancy)
	52 BT656
	53 MP1S
	54 MP2P
	55 BMPEG
	255 Invalid

0x2A0A Media Payload Size

SIP

Used in:
0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2A0A
2, 3	Length 0x0002
4	Value Packet Duration in milliseconds

0x2A0B Media Clock Rate

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2A0B
2, 3	Length 0x0004
4-7	Value[0-3] Clock Rate

0x2A0E Media Connection Address

This TLV provides the IP address for each media stream.

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2A0E
2, 3	Length 0x0004
4-7	Value[0-3] IP Address

0x2A13 Media Flow Direction

This TLV defines the media flow direction.

Used in:

0x0033 NPDI Universal ICB

Byte	Description
0, 1	Tag 0x2A13
2, 3	Length 0x0001
4	Value 0xFF Invalid 0x00 Send and Receive (generates SDP line a=sendrecv) 0x01 Receive Only (generates SDP line a=recvonly) 0x02 Send Only (generates SDP line a=sendonly)

0x2A17 SDP Media Stream Native Text

Used in:

0x0033 NPDI Universal ICB in:

PPL Event Request message

Byte	Description
0, 1	Tag 0x2A17
2, 3	Length Variable
4-n	Value Raw SDP text (not null terminated)

0x4E21 Message Configuration Valid

Used in:

SS7 ISUP Message Format Configure message

Byte	Description
0, 1	Tag 0x4E21
2, 3	Length 0x0001
4	Value[0] 0x00 Invalid 0x01 Valid (Default)

0x4E22 Message Processing Supported

The following table defines the format of the Message Configuration Supported TLV:

Used in:

SS7 ISUP Message Format Configure message

Byte	Description
0, 1	Tag 0x4E22
2, 3	Length 0x0001
4	Value[0] 0x00 Not Supported 0x01 Supported (Default)

0x4E24 Acceptance Rate

Used in:

0x1E Generic PPL ICB in *PPL Request* message

Byte	Description
0, 1	Tag 0x4E24
2, 3	Length 0x0003
4	Value[0] ACL Level 0x01 or 0x02
5,6	Value[1,2] Acceptance Rate - Number of calls per second for this ACL Level. MSB - Acceptance Rate LSB - Acceptance Rate

0x9041 Protocol Parameters

Use this TLV to either:

- assign the default standard and network identity

or

- create maps and assign the values for standard and network identity to the maps

Used in:

NGA Configure message

Byte	Description
0, 1	Tag 0x9041
2, 3	Length 0x0015

4-7	Value[0-3] Default Standard 0x00 - M3UA Stack ANSI 0x01 - M3UA Stack ITU
8-11	Value [4-7] Default Network Identity 0x00 - M3UA NW International 0x01 - M3UA NW National No. Priority 0x02 - M3UA NW National Priority 0x03 - M3UA NW Invalid
12	Value [8] Number of Maps
13-16	Value[9-12] Network Appearance 4-byte number unique in the network
17-20	Value[13-16] Standard 0x00 - M3UA Stack ANSI 0x01 - M3UA Stack ITU
21-24	Value [17-20] Network Identity 0x00 - M3UA NW International 0x01 - M3UA NW National No. Priority 0x02 - M3UA NW National Priority 0x03 - M3UA NW Invalid

0x9042 Application Server

Used in:

NGA Configure message

Byte	Description
0, 1	Tag 0x9042
2, 3	Length 0x0025
4-7	Value[0-3]Traffic Mode 0x01- M3UA AS Mode Override
8-11	Value[4-7] Routing Key Type 0x00 - DPC 0x01 - DPC SIO 0x02 - DPC SSN 0x03 - DPC CIC SIO 0x04 - DPC OPC 0x05 - DPC OPC CIC
12-15	Value[8-11] Point Code Number
16-19	Value[12-15] Network Appearance Number

20	Value[16] Number of Ranges
21	Value[17] Range OPC SIO SSN Low CIC
22	Value[18] Number of ASPs
23	Value[19] ASP ID
24	Value[20] Number of routing contexts
25-28	Value[21-24] Routing contexts Routing Context ID Signaling Gateway ID

0x9043 Application Server Process

Used in:
NGA Configure message

Byte	Description
0, 1	Tag 0x9043
2, 3	Length Variable
4-n	Value[0,1] Primary IP Address
:	Value[2, 3] Secondary IP Address

0x9044 Signaling Gateway

Used in:
NGA Configure message

Byte	Description
0, 1	Tag 0x9044
2, 3	Length 0x0009
4-7	Value[0-3] Traffic Mode 0x00000001 - M3UA AS Mode Override 0x00000003 - M3UA AS Mode Broadcast
8-9	Value[4,5] Primary Signaling Gateway Process ID
10	Value[6] Number of Signaling Gateway Processes
11-12	Value[7,8] Signaling Gateway Process ID

0x9045 Signaling Gateway Process

Used in:

NGA Configure message

Byte	Description
0, 1	Tag 0x9045
2, 3	Length 0x0007
4,5	Value[0,1] Port Number
6	Value [2] Number of IP Addresses
7-10	Value [3-6] IP Addresses

0x9046 Connection

Used in:

NGA Configure message

Byte	Description
0, 1	Tag 0x9046
2, 3	Length 0x0004
4,5	Value[0,1] Application Server Process ID
6,7	Value[2,3] Signaling Gateway Process ID

0x9048 Application Server In Service

Used in:

NGA Service Configure message

Byte	Description
0, 1	Tag 0x9048
2, 3	Length 0x0004
4	Value[0] Number of Logical Connections
5	Value[1] Logical Connection ID

0x9049 Route Set Query

Used in:

NGA Configure Query message

Byte	Description
0, 1	Tag 0x9049
2, 3	Length 0x0004
4,5	Value[0,1] Number of Logical Connections
6,7	Value[2,3] Logical Connection ID

0x904A Protocol Query

Used in:

NGA Configure Query message

Byte	Description
0, 1	Tag 0x904A
2, 3	Length 0x0024
4,5	Value[0,1] Maximum Application Service Provider
6,7	Value[2,3] Maximum Application Server
8,9	Value[4,5] Maximum Signaling Gateway Process
10,11	Value[6,7] Maximum Signaling Gateway
12-15	Value[8-11] Maximum Route Sets
16-19	Value [12-15] Default Standard
20-23	Value[16-19] Default Network identify
24,25	Value[20,21] Maximum User Parts
26,27	Value[22,23] Maximum Connections
28	Value [24] Number of Maps
29-32	Value [25-28] Network Appearance
33-36	Value[29-32] Standard
37-40	Value[33-36] Network Identity

0x904B Application Server Query

Used in:

NGA Configure Query message

Byte	Description
0, 1	Tag 0x904F
2, 3	Length 0x0006
4, 5	Value[0,1] Maximum Number of Application Servers
6, 7	Value[2,3] Actual Number of Application Servers
8, 9	Value[4,5] Application Server ID

0x904C Application Server Process Query

Used in:

NGA Configure Query message

Byte	Description
0, 1	Tag 0x904C

2, 3	Length Variable
4,5	Value[0,1] Port
6	Value[2] Number of IP addresses
7-10	Value[3-6] IP Address (4 bytes)
:	Value [:]
:	Value[:] IP Address (4 bytes)

0x904D Signaling Gateway Query

Used in:

NGA Configure Query message

Byte	Description
0, 1	Tag 0x904D
2, 3	Length 0x0003
4	Value[0] Mode 0x01 - M3UA AS Mode Override 0x03 - M3UA AS Mode Broadcast
5	Value[1] Number of Signaling Gateway Processes
6	Value[2] Signaling Gateway Process ID

0x904E Signaling Gateway Process Query

Used in:

NGA Configure Query message

Byte	Description
0, 1	Tag 0x904E
2, 3	Length Variable
4,5	Value[0] Port
6	Value[2] Number of IP addresses
7-10	Value[3-6] IP Address (4 bytes)
:	Value [:]
:	Value[:] IP Address (4 bytes)

0x904F Connection Query

Used in:

NGA Configure Query message

Byte	Description
0, 1	Tag 0x904F
2, 3	Length 0x0004
4,5	Value[0,1] Application Server Process ID
6,7	Value[2,3] Signaling Gateway Process ID

0x9050 General Query Data

Used in:

NGA Configure Query message

Byte	Description
0, 1	Tag 0x9050
2, 3	Length 0x0006
4, 5	Value[0,1] Maximum number of objects
6, 7	Value[2,3] Actual number of objects
8, 9	Value[4,5] Object ID

0x9051 Application Server State Query

Used in:

NGA Configure Query message*NGA State Notify* message*NGA Service Query* message

Byte	Description
0, 1	Tag 0x9051
2, 3	Length 0x0006
4	Value[0] Application Service State 0x00 - Out of Service 0x01 - In Service
5, 6	Value[1,2] Number of logical connections
7, 8	Value[3,4] Logical Connection ID
9	Value[5] Logical Connection State 0x00 - Out of Service 0x01 - In Service

0x9052 Route Set

Used in:

NGA Configure message

Byte	Description
0,1	Tag 0x9052
2,3	Length 0x0010
4-7	Value[0-3] Destination Point Code
8,9	Value[4,5] Number of Routes
10,11	Value[6,7] Route ID
12,13	Value[8,9] Signaling Gateway ID
14-17	Value[10-13] Network Appearance
18,19	Value [14,15] Priority

0x9053 Modify Destination

Used in:

NGA Configure message

Byte	Description
0, 1	Tag 0x9053
2, 3	Length 0x0010
4-7	Value[0-3] Modify Option 0x00000000 - Add Route 0x00000001 - Remove Route
8,9	Value[4,5] Number of Routes
10,11	Value[6,7] Route ID
12,13	Value[8,9] Signaling Gateway ID
14-17	Value[10-13] Network Appearance
16,17	Value [14,15] Priority

0x9054 Modify Protocol

Used in:

NGA Configure message

Byte	Description
0, 1	Tag 0x9054
2,3	Length 0x0011
4-7	Value[0-3] Modify Option 0x00000000 - Add Route 0x00000001 - Remove Route
8	Value[4] Number of Maps
9-12	Value[6] Network Appearance
13-16	Value [7] Standard 0x00000000 - ANSI 0x00000001 - ITU
17-20	Value [8] Network Identity 0x00000000 - International 0x00000001 - National Number Priority 0x00000002 - National Priority 0x00000003 - Invalid Network

0xE001 Command

Used to supports M3UA functionality.

Used in:

NGA Configure message

NGA Service State Configure message

Byte	Description
0, 1	Tag 0xE001
2, 3	Length 0x0002
4, 5	Value[0,1] 0x0000 - Add 0x0001 - Remove 0x0002 - Modify 0x0003 - Up 0x0004 - Down 0x0005 - In Service 0x0006 - Out of Service

0xE002 Service State

Used to supports M3UA functionality.

Used in:

NGA State Notify message

Byte	Description
0, 1	Tag 0xE002
2, 3	Length 0x0002
4, 5	Value[0,1] 0x0005 - Out of Service 0x0006 - In Service

0xE003 AIB Container TLV

Used in the Resource Delete Indication message (0x0129) when deleting a child conference

Byte	Description
0, 1	Tag 0xE003
2, 3	Length 0x0008

4	Value[0]	Address Method 0x00 (Single Entity)
5	Value[1]	Address Element Count 0x01
6	Value[2]	Address Element 0x45
7	Value[3]	Data Length 0x04
8, 9	Value[4, 5]	Parent Conference ID
10, 11	Value[6, 7]	Child Conference ID

BT IUP TLVs

Overview The table below shows the format of the TLVs used in the BT IUP ICB for BT IUP messages.

TLV Name	TLV ID	TLV Length	TLV Data																			
			Byte Number	Bit Description																		
H0, H1 Element	0x01	0x02	0x01	The H0 value																		
			0x02	The H1 value																		
IAM/IFAM Message Indicators	0x02	0x05	0x01	<table border="0"> <tr> <td><u>Bit</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0-5</td> <td>Calling Party Category (CPC) or (CDPC) Only bits 0 through 5 are valid. (The values are different for CPC and CDPC usage.)</td> </tr> <tr> <td>6</td> <td>Enhanced Diversion Indicator (EDI)</td> </tr> <tr> <td>7</td> <td>0</td> </tr> </table>	<u>Bit</u>	<u>Definition</u>	0-5	Calling Party Category (CPC) or (CDPC) Only bits 0 through 5 are valid. (The values are different for CPC and CDPC usage.)	6	Enhanced Diversion Indicator (EDI)	7	0										
			<u>Bit</u>	<u>Definition</u>																		
0-5	Calling Party Category (CPC) or (CDPC) Only bits 0 through 5 are valid. (The values are different for CPC and CDPC usage.)																					
6	Enhanced Diversion Indicator (EDI)																					
7	0																					
			0x02	<table border="0"> <tr> <td><u>Bit</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0</td> <td>Calling Line Identity Indicator (CLI)</td> </tr> <tr> <td>1</td> <td>CLI Blocking Indicator (CBI)</td> </tr> <tr> <td>2</td> <td>International Indicator (INT)</td> </tr> <tr> <td>3</td> <td>Interworking Indicator (I/W)</td> </tr> <tr> <td>4</td> <td>Priority Access Indicator (PA)</td> </tr> <tr> <td>5</td> <td>0</td> </tr> <tr> <td>6</td> <td>Meter Delay Guard Timeout Indicator (MDG)</td> </tr> <tr> <td>7</td> <td>Protection Indicator (PROT)</td> </tr> </table>	<u>Bit</u>	<u>Definition</u>	0	Calling Line Identity Indicator (CLI)	1	CLI Blocking Indicator (CBI)	2	International Indicator (INT)	3	Interworking Indicator (I/W)	4	Priority Access Indicator (PA)	5	0	6	Meter Delay Guard Timeout Indicator (MDG)	7	Protection Indicator (PROT)
<u>Bit</u>	<u>Definition</u>																					
0	Calling Line Identity Indicator (CLI)																					
1	CLI Blocking Indicator (CBI)																					
2	International Indicator (INT)																					
3	Interworking Indicator (I/W)																					
4	Priority Access Indicator (PA)																					
5	0																					
6	Meter Delay Guard Timeout Indicator (MDG)																					
7	Protection Indicator (PROT)																					
IAM/IFAM Message Indicators	0x02	0x05	0x03	<table border="0"> <tr> <td><u>Bits</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0-3</td> <td>Service Handling Protocol (SHP)- (Currently only 0 and 1 are supported.)</td> </tr> <tr> <td>4</td> <td>Release Protocol Indicator (RPI)</td> </tr> <tr> <td>5</td> <td>Long Propagation Delay Indicator (LPD)</td> </tr> <tr> <td>6</td> <td>Reserved</td> </tr> <tr> <td>7</td> <td>Call Type Indicator (CTI) – L bit</td> </tr> </table>	<u>Bits</u>	<u>Definition</u>	0-3	Service Handling Protocol (SHP)- (Currently only 0 and 1 are supported.)	4	Release Protocol Indicator (RPI)	5	Long Propagation Delay Indicator (LPD)	6	Reserved	7	Call Type Indicator (CTI) – L bit						
			<u>Bits</u>	<u>Definition</u>																		
			0-3	Service Handling Protocol (SHP)- (Currently only 0 and 1 are supported.)																		
4	Release Protocol Indicator (RPI)																					
5	Long Propagation Delay Indicator (LPD)																					
6	Reserved																					
7	Call Type Indicator (CTI) – L bit																					
			0x04	<table border="0"> <tr> <td><u>Bits</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0-1</td> <td>CTI M and N bits</td> </tr> <tr> <td>2</td> <td>Echo Control Device Indicator (ECD)</td> </tr> <tr> <td>3</td> <td>Network Translated Address Indicator (NTA)</td> </tr> <tr> <td>4-7</td> <td>Interconnect Specific Information</td> </tr> </table>	<u>Bits</u>	<u>Definition</u>	0-1	CTI M and N bits	2	Echo Control Device Indicator (ECD)	3	Network Translated Address Indicator (NTA)	4-7	Interconnect Specific Information								
<u>Bits</u>	<u>Definition</u>																					
0-1	CTI M and N bits																					
2	Echo Control Device Indicator (ECD)																					
3	Network Translated Address Indicator (NTA)																					
4-7	Interconnect Specific Information																					
			0x05	<table border="0"> <tr> <td><u>Bits</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0-3</td> <td>Routing Control Indicator (RCI)</td> </tr> <tr> <td>4-6</td> <td>Call Path Indicator (CPI) (Currently only 0 and 2 are supported.)</td> </tr> <tr> <td>7</td> <td>Presentation Number Indicator (PNI)</td> </tr> </table>	<u>Bits</u>	<u>Definition</u>	0-3	Routing Control Indicator (RCI)	4-6	Call Path Indicator (CPI) (Currently only 0 and 2 are supported.)	7	Presentation Number Indicator (PNI)										
<u>Bits</u>	<u>Definition</u>																					
0-3	Routing Control Indicator (RCI)																					
4-6	Call Path Indicator (CPI) (Currently only 0 and 2 are supported.)																					
7	Presentation Number Indicator (PNI)																					

TLV Name	TLV ID	TLV Length	TLV Data											
			Byte Number	Bit Description										
Called Address	0x03	0x01-0x0B	0x01	<table border="0"> <tr> <td><u>Bits</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0-4</td> <td>Number of Called Address Signals</td> </tr> <tr> <td>5-7</td> <td>PRI</td> </tr> </table>	<u>Bits</u>	<u>Definition</u>	0-4	Number of Called Address Signals	5-7	PRI				
			<u>Bits</u>	<u>Definition</u>										
0-4	Number of Called Address Signals													
5-7	PRI													
			0x02-0x0B	From 1-10 bytes of BCD information may be present.										
Line Identity	0x04	0x01-0x09	0x01	<table border="0"> <tr> <td><u>Bits</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0-1</td> <td>BA-Bits Nature of Address Indicator (NAI)</td> </tr> <tr> <td>2</td> <td>Line Identity C-Bit, Incomplete Address Indicator (IAI)</td> </tr> <tr> <td>3</td> <td>Line Identity D-Bit, Identity Qualified (IQ)</td> </tr> <tr> <td>4-7</td> <td>Number of Address Signals</td> </tr> </table>	<u>Bits</u>	<u>Definition</u>	0-1	BA-Bits Nature of Address Indicator (NAI)	2	Line Identity C-Bit, Incomplete Address Indicator (IAI)	3	Line Identity D-Bit, Identity Qualified (IQ)	4-7	Number of Address Signals
			<u>Bits</u>	<u>Definition</u>										
0-1	BA-Bits Nature of Address Indicator (NAI)													
2	Line Identity C-Bit, Incomplete Address Indicator (IAI)													
3	Line Identity D-Bit, Identity Qualified (IQ)													
4-7	Number of Address Signals													
			0x02-0x09	From 1-8 bytes of BCD information may be present.										
ASUI Message Information Contained Code (ICC)	0x05	0x01	0x01	All bits are valid.										
Partial Calling Line Identity (PCLI)	0x06	0x09	0x01	<table border="0"> <tr> <td><u>Bits</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0-3</td> <td>D1 Type of Switch</td> </tr> <tr> <td>4-7</td> <td>D2 Type of Switch</td> </tr> </table>	<u>Bits</u>	<u>Definition</u>	0-3	D1 Type of Switch	4-7	D2 Type of Switch				
			<u>Bits</u>	<u>Definition</u>										
			0-3	D1 Type of Switch										
			4-7	D2 Type of Switch										
			0x02	<table border="0"> <tr> <td><u>Bits</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0-3</td> <td>D1 PNO Identity</td> </tr> <tr> <td>4-7</td> <td>D2 PNO Identity</td> </tr> </table>	<u>Bits</u>	<u>Definition</u>	0-3	D1 PNO Identity	4-7	D2 PNO Identity				
			<u>Bits</u>	<u>Definition</u>										
0-3	D1 PNO Identity													
4-7	D2 PNO Identity													
0x03	<table border="0"> <tr> <td><u>Bits</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0-3</td> <td>D3 PNO Identity</td> </tr> <tr> <td>4-7</td> <td>D1 Switch Number</td> </tr> </table>	<u>Bits</u>	<u>Definition</u>	0-3	D3 PNO Identity	4-7	D1 Switch Number							
<u>Bits</u>	<u>Definition</u>													
0-3	D3 PNO Identity													
4-7	D1 Switch Number													
0x04	<table border="0"> <tr> <td><u>Bits</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0-3</td> <td>D2 Switch Number</td> </tr> <tr> <td>4-7</td> <td>D3 Switch Number</td> </tr> </table>	<u>Bits</u>	<u>Definition</u>	0-3	D2 Switch Number	4-7	D3 Switch Number							
<u>Bits</u>	<u>Definition</u>													
0-3	D2 Switch Number													
4-7	D3 Switch Number													
0x05	Bilateral Agreement Byte 1													
0x06	Bilateral Agreement Byte 2													
Partial Calling Line Identity (PCLI) continued	0x06	0x09	0x07	Bilateral Agreement Byte 3										
			0x08	Bilateral Agreement Byte 4										
			0x09	Bilateral Agreement Byte 5										
Number of Digits Requested	0x07	0x01	0x01	Only bits 0 through 3 are valid.										
SAUSI Information Indicator	0x08	0x01	0x01	All bits are valid.										

TLV Name	TLV ID	TLV Length	TLV Data								
			Byte Number	Bit Description							
Address Complete Message Indicators	0x09	0x03	0x01	<table border="0"> <tr> <td><u>Bits</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0-5</td> <td>Charge Indicator</td> </tr> <tr> <td>6-7</td> <td>Unused</td> </tr> </table>	<u>Bits</u>	<u>Definition</u>	0-5	Charge Indicator	6-7	Unused	
			<u>Bits</u>	<u>Definition</u>							
			0-5	Charge Indicator							
6-7	Unused										
0x02	<table border="0"> <tr> <td><u>Bits</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0-5</td> <td>Unused</td> </tr> <tr> <td>6</td> <td>Last Party Release Indicator (LPR)</td> </tr> <tr> <td>7</td> <td>Unused</td> </tr> </table>	<u>Bits</u>	<u>Definition</u>	0-5	Unused	6	Last Party Release Indicator (LPR)	7	Unused		
<u>Bits</u>	<u>Definition</u>										
0-5	Unused										
6	Last Party Release Indicator (LPR)										
7	Unused										
0x03	<table border="0"> <tr> <td><u>Bits</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0</td> <td>Last Party Release Indicator</td> </tr> <tr> <td>1-5</td> <td>Interworking Indicators</td> </tr> <tr> <td>6</td> <td>Echo Control Device Indicator</td> </tr> <tr> <td>7</td> <td>Reserved</td> </tr> </table>	<u>Bits</u>	<u>Definition</u>	0	Last Party Release Indicator	1-5	Interworking Indicators	6	Echo Control Device Indicator	7	Reserved
<u>Bits</u>	<u>Definition</u>										
0	Last Party Release Indicator										
1-5	Interworking Indicators										
6	Echo Control Device Indicator										
7	Reserved										
CNA Reason	0x0A	0x01	0x01	All bits are valid.							
CNA Diagnostics	0x0B	0x01	0x01	All bits are valid.							
Type of Answer	0x0C	0x01	0x01	Only bits 0 through 3 are valid.							
Release Reason	0x0D	0x01	0x01	All bits are valid.							
Confusion Message Qualifier	0x0E	0x01	0x01	All bits are valid.							
ACI Information Contained and Information Requested	0x0F	0x02	0x01	Information Contained Code (ICC)							
			0x02	Information Requested Code (IRC)							
ACI Type 1 Reserved Bytes	0x10	0x02	0x01	Reserved Byte 1							
			0x02	Reserved byte 2							
ACI Type 1 Message Indicators	0x11	0x01	0x01	<table border="0"> <tr> <td><u>Bits</u></td> <td><u>Definition</u></td> </tr> <tr> <td>0-6</td> <td>Unused</td> </tr> <tr> <td>7</td> <td>Operator Indicator</td> </tr> </table>	<u>Bits</u>	<u>Definition</u>	0-6	Unused	7	Operator Indicator	
<u>Bits</u>	<u>Definition</u>										
0-6	Unused										
7	Operator Indicator										

TLV Name	TLV ID	TLV Length	TLV Data		
			Byte Number	Bit Description	
Calling/Called Subscriber's Basic Service Marks	0x12	0x02	0x01	<u>Bits</u>	<u>Definition</u>
				0	A - Admin/Maintenance Call Barring Indicator
				1	B - Subscriber Controlled Incoming Calls Barred Indicator
				2	C - Pre-arranged Incoming Calls Barred Indicator
				3	D - Permanent Incoming Calls Barred Indicator
				4	E - Temporary Out of Service (TOS)
				5	F - ICB, Except for Operator, Indicator
				6	G - Called Subscriber Facility Information Indicator
Calling/Called Subscriber's Basic Service Marks Continued	0x12	0x02	0x02	<u>Bits</u>	<u>Definition</u>
				0	I - Permanent Outgoing Calls Barred Indicator
				1	J - Outgoing Local Calls Barred Indicator
				2	K - Outgoing National Calls Barred Indicator
				3	L - Outgoing International Calls Barred Indicator
				4	M - Operator Calls Barred Indicator
				5	N - Supplementary Facility Calls Barred Indicator
				6	O - Digit Masking Indicator
ACI Type 3 Message Indicators	0x13	0x01	0x01	<u>Bits</u>	<u>Definition</u>
				0-5	Calling/Called Subscriber's Tariff Group
				6	Reserved
ACI Type 4 Message Indicators	0x14	0x01	0x01	<u>Bits</u>	<u>Definition</u>
				0-5	Called Subscriber's Tariff Group
				6-7	Reserved

TLV Name	TLV ID	TLV Length	TLV Data															
			Byte Number	Bit Description														
Calling Subscriber's Originating Facility Marks	0x15	0x02	0x01	<table border="1"> <thead> <tr> <th>Bits</th> <th>Definition</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A - Disabled Subscriber's Indicator</td> </tr> <tr> <td>1</td> <td>B - Attendant Call Office Indicator</td> </tr> <tr> <td>2</td> <td>C - Advise Duration and Charge (AD&C) Indicator</td> </tr> <tr> <td>3</td> <td>D - PBX Subscriber Indicator</td> </tr> <tr> <td>4-7</td> <td>Reserved</td> </tr> </tbody> </table>	Bits	Definition	0	A - Disabled Subscriber's Indicator	1	B - Attendant Call Office Indicator	2	C - Advise Duration and Charge (AD&C) Indicator	3	D - PBX Subscriber Indicator	4-7	Reserved		
			Bits	Definition														
0	A - Disabled Subscriber's Indicator																	
1	B - Attendant Call Office Indicator																	
2	C - Advise Duration and Charge (AD&C) Indicator																	
3	D - PBX Subscriber Indicator																	
4-7	Reserved																	
0x02	Reserved																	
Called Subscriber's Terminating Facility Marks	0x16	0x02	0x01	<table border="1"> <thead> <tr> <th>Bits</th> <th>Definition</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A - SVI Indicator</td> </tr> <tr> <td>1</td> <td>B - CNI Indicator</td> </tr> <tr> <td>2</td> <td>C - PBX Night Interception Indicator</td> </tr> <tr> <td>3</td> <td>D - Call Waiting Indicator</td> </tr> <tr> <td>4</td> <td>E - Fixed Destination Service Indicator</td> </tr> <tr> <td>5-7</td> <td>Reserved</td> </tr> </tbody> </table>	Bits	Definition	0	A - SVI Indicator	1	B - CNI Indicator	2	C - PBX Night Interception Indicator	3	D - Call Waiting Indicator	4	E - Fixed Destination Service Indicator	5-7	Reserved
			Bits	Definition														
0	A - SVI Indicator																	
1	B - CNI Indicator																	
2	C - PBX Night Interception Indicator																	
3	D - Call Waiting Indicator																	
4	E - Fixed Destination Service Indicator																	
5-7	Reserved																	
0x02	Reserved																	
Generic TLC Entry (When this TLV occurs it must be the only TLV used. It must contain the entire user-defined message.)	0x17	0x01-0xC8	0x01-0xC8	First byte of generic TLV Entry Data. From 1-200 bytes of generic data may be present in this TLV.														
Reserved Data Byte	0x18	0x01	0x01	Reserved														
SAM and FAM Called Address	0x19	0x01-0x0A	0x01	<table border="1"> <thead> <tr> <th>Bits</th> <th>Definition</th> </tr> </thead> <tbody> <tr> <td>0-4</td> <td>Number of Called Address Signals</td> </tr> <tr> <td>5-7</td> <td>PRI</td> </tr> </tbody> </table>	Bits	Definition	0-4	Number of Called Address Signals	5-7	PRI								
			Bits	Definition														
0-4	Number of Called Address Signals																	
5-7	PRI																	
0x02-0x0A	From 1-9 bytes of BCD information may be present.																	
Accumulated Called Digits (This TLV is only sent from the switch to the host. It will be present whenever an IAM or IFAM H0,H1 TLV is present within an IUP ICB.)	0x1A	0x01-0x0F	0x01	Number of Called Digits to follow. Packed two BCD digits per byte.														
			0x02-0x0F	Bytes of BCD digits.														
ISDN Composite SIM - Facility Indicator Code (FIC)	0x1B	0x02	0x01	LSB - Facility Indicator Code														
			0x02	MSB - Facility Indicator Code														

TLV Name	TLV ID	TLV Length	TLV Data	
			Byte Number	Bit Description
Closed User Group (CUG) Interlock Code	0x1C	0x04	0x01	<u>Bits</u> <u>Definition</u> 0-3 D1 6-7 D2
			0x02	<u>Bits</u> <u>Definition</u> 0-3 D3 4-7 D4
			0x03	LSB - Binary Code
			0x04	MSB - Binary Code
Service Indicator Code	0x1D	0x02	0x01	First Octet
			0x02	Second Octet
Six Character Network Address Extension (NAE) - Subaddress	0x1E	0x02-0x07	0x01	NAE Character Count (range 1 to 6)
			0x02	1st NAE Character
			n	nth NAE Character
Service Code	0x1F	0x01	0x01	Service Code

Mapping BT IUP Messages to TLV Format

This section defines the TLVs required in a IUP DATA ICB to compose each message. TLVs may appear in any order within the IUP DATA ICB. If a TLV is not included and is required for the underlying PNO-ISC IUP Message a default value will be supplied from the appropriate configuration byte. Therefore, many messages can be generated from the configuration data area by only encoding a message containing the HOH1 TLV, since this uniquely identifies the message to be sent. However, several messages require additional TLVs to uniquely identify the message (for example, ACI and ASUI).

BT IUP Message	Valid TLVs
Initial Address Message/ Final Address Message (IAM/IFAM)	0x01, 0x02, 0x03, 0x04
Subsequent Address Message/ Final Address Message (SAM/FAM)	0x01, 0x18, 0x19
Additional Setup Information, Type 1 - Full Calling Line Identity	0x01, 0x04, 0x05
Additional Setup Information, Type 2 - Partial Calling Line Identity	0x01, 0x05, 0x06, 0x18
Send N Digits	0x01, 0x07
Send All Digits	0x01, 0x18
Send Additional Setup Information	0x01, 0x08
Address Complete	0x01, 0x09
Congestion	0x01
Terminal Congestion	0x01
Connection Not Admitted	0x01, 0x0A, 0x0B
Repeat Attempt	0x01
Subscriber Engaged	0x01
Subscriber Out of Order	0x01
Answer	0x01, 0x0C
Clear	0x01
Release	0x01, 0x0D
Circuit Free	0x01
Blocking	0x01
Unblocking	0x01
Blocking Acknowledgement	0x01
Unblocking Acknowledgement	0x01
Overload	0x01
Confusion	0x01

BT IUP Message	Valid TLVs
Additional Call Information, Type 1	0x01 , 0x04, 0x0F , 0x10, 0x11
Additional Call Information, Type 2	0x01 , 0x0F , 0x10
Additional Call Information, Type 3	0x01 , 0x04, 0x0F , 0x12, 0x13
Additional Call Information, Type 4	0x01 , 0x0F , 0x12, 0x14
Additional Call Information, Type 5	0x01 , 0x0F , 0x15
Additional Call Information, Type 6	0x01 , 0x0F , 0x16
Additional Call Information, Type 7	0x01 , 0x0F

5 Response Status Values

Overview

Purpose This chapter lists the Response Status Values (RSVs) used in the EXS API for the CSP. These values are in hexadecimal.

Response Status Values List

Hex ID	Name	Description as necessary
00 00-00 0F	These values differ depending on the message that returns them. Please refer to the specific message for the response status values.	
00 10	Positive Acknowledgment (ACK)	The message was successfully processed, and any required actions were successfully completed.
00 11	Bad Checksum	The message checksum does not match the checksum calculated by the system. An error must exist or a field must be missing.
00 12	Invalid Logical Span ID	A field reserved for a logical span ID contains a number larger than the capacity of the switching matrix.
00 13	Invalid Channel Number	A field reserved for a channel number is outside the range of 0–30 for E1 or 0–23 for T1. For E1, you can reference channel 30 only if the channel is configured for CCS, with the <i>E1 Span Configure</i> message.
00 14	Message Invalid for Current Matrix State	The CSP Matrix Series 3 Card is not in a state to accept this message.
00 15	Negative Acknowledgment (NACK)	Message could not be successfully processed. The span in which a link is assigned, can have either CICs or not used at all.
00 16	Invalid Address Range	The address range specified in a block configuration message is not valid.
00 17	Invalid Data Type	Returned in response to a message with invalid data parameters.
00 18	Invalid Channel B State	Channel B is in an invalid state for the action required. The <i>More Status</i> field indicates the state. NOTE: This does not pertain to ISDN B channels.
00 19	Trunk Status Dead	One of the parties involved in a Call Processing message is associated with a dead line.
00 1A	Inseize Failure	Inseize could not be successfully completed (for example, distant end signaling error).
00 1B	Outseize Failure, No Acknowledgment	Outseize failed because acknowledgment signaling conditions were never met (for example, a channel of E&M wink start trunk did not receive a wink).
00 1C	Outseize Failure, Glare	Outseize attempted on trunk involved in inseizure processing.
00 1D	Invalid Channel A State or Invalid PPL Event	Channel A is in an invalid state for the action required. The <i>Status</i> field indicates the state of Channel A. A PPL component has received an invalid event for the current PPL state.
00 1E	Memory Allocation Failure	Channel configuration change failed due to lack of available memory.
00 1F	Invalid Group Number	Reserved for future use.
00 20	Invalid Action Value	Invalid decision parameter action value was received.
00 21	Invalid Time	The value entered for the time is not valid.
00 22	First Outseize Instruction Not A Seize	The first instruction in an <i>Outseize Control</i> message must be a Seize if the channel is idle.
00 23	Outseize "Seize" Instruction Not Allowed	This response is returned if one of the following occurs: An <i>Outseize Instruction List Configure</i> message is sent with a Seize instruction (a Seize instruction cannot be preprogrammed). An <i>Outseize Control</i> message with a Seize instruction is sent when one has been sent previously and the call is still active.

00 24	Outseize Failed Due To No Answer	Outseize failed due to no answer from the distant end.
00 25	Invalid Encoding Format	One of the following is invalid: the encoding format of RAN or the encoding format in a <i>Conference Create</i> message.
00 26	Invalid Conference Size	Sent in response to a host-initiated <i>Conference Create</i> message, when the conference size indicated is less than 2 or greater than the maximum number of channels allowed to be conferenced together.
00 27	DSP Not Configured for Requested Function	No DSPs are configured for the function requested by the host (for example, configuration for service).
00 28	DSP Resources Not Available	No DSP resources are currently available for the requested DSP function. Applies to DSP functions which are not managed as shared resources, for example, <i>Conference Create</i> .
00 29	Invalid Conference Output Option	Conference output option indicated in the <i>Conference Create</i> message is not valid.
00 2A	Invalid Conference ID	Conference ID sent by the host is not associated with any conferences previously established through the <i>Conference Create</i> message. or The conference associated with this conference ID may have been marked for deletion.
00 2B	<i>Conference Create</i> Failure	Conference creation failed while the conference was establishing (for example, the MFDSP card was removed).
00 2C	Conference Establishing	Returned in a <i>Delete Conference</i> , <i>Connect to Conference</i> , or <i>Connect 1-Way to Conference</i> message if the conference associated with the conference ID is in the process of being created.
00 2D	Incompatible PCM Encoding for Conference	Returned in a <i>Connect to Conference</i> and <i>Connect 1-Way to Conference</i> message when the channel encoding format doesn't match that of the conference (for example, A-law encoded channel connected to a μ -law encoded conference).
00 2E	Conference Size Exceeded	Returned in a <i>Connect to Conference</i> message when the conference size has been exceeded.
00 2F	Maximum Broadcast Number Exceeded	Returned in a <i>Connect 1-Way</i> or <i>Connect 1-Way to Conference</i> message when the number of 1-Way broadcasts exceeds the maximum allowed.
00 30	DSP Resource Already Allocated	Returned in a <i>Collect Digit String</i> or <i>DSP Service Request</i> message if digit collection is already active on the specified channel.
00 31	Channel Violation	Returned if an CSP 1000 Matrix Card card is being used in a system with more than 1,024 channels.
00 32	Channels A and B Already Connected	A connect request has been sent for two channels which are already connected.
00 33	SS7 Configuration Error	Possible scenarios regarding virtual spans: 1. Configuring virtual CICs on remote nodes using an SS7 PQ card in the server node. 2. Configuring virtual CICs on a virtual VOCC card (type 0x80) using either an SS7 PQ card or an HP card. 3. Configuring virtual CICs beyond five virtual T-ONE cards or four virtual E-ONE cards on the server node using an SS7 PQ card.
00 34	Ring Timeslot Inaccessible	Remote timeslot is not accessible from this node. Check CSP hardware.
00 35	Invalid Logical Ring ID	
00 36	Unassigned Ring	
00 37	Ring Already Assigned	
00 38	Invalid Ring State	

00 39	No Bandwidth Available	
00 3D	Invalid ICB Data	An information control block has invalid length or count field, or the data within the ICB is invalid. Invalid TLV Count Invalid TLV Length
00 3E	ISDN Congestion	
00 40	SRecord Invalid	
00 41	Download aborted due to invalid SRecord.	
00 42	Restart download due to sequence problem.	
00 44	BRecord Invalid	BRecord not accepted due to invalid value.
00 43	Node Not Ready	A valid Logical Node ID has not been assigned (therefore, <i>Polls</i> cannot be requested) or the CSP is not in a valid state for the requested action (such as <i>Download Begin</i> during Boot State).
00 44	No Download Begin Message Received	A <i>Download SRecord</i> or <i>Download BRecord</i> message was sent without a preceding <i>Download Begin SRecord</i> or <i>Download Begin BRecord</i> message.
00 45	Invalid Download	Sent in response to a <i>Download Complete</i> message when the download does not validate due to an incorrect checksum.
00 47	Invalid Ring Loop Timing	
00 4B	RTP Port in Use	
00 54	SS7 Invalid SCCP TCAP Parameter	An SS7 SCCP TCAP configuration attempt specified an invalid SS7 SCCP TCAP parameter.
00 55	SS7 Invalid SCCP TCAP Configuration Parameter	An SS7 SCCP TCAP configuration attempt specified an invalid SS7 SCCP TCAP configuration parameter. A CSP returns this indication when a host queries for a configuration parameter using the message, SS7 SCCP/TCAP Query (0x78).
00 56	System Busy Condition on Remote Node	
00 57	Conferencing Disabled	
00 58	Invalid Remote Timeslot for Connect 1-way Forced	B address is not a valid remote timeslot. For SIP calls, B address is not bearer switched.
00 59	Outseize Failure, Blocked	
00 5A	DS3 Out of Service	An attempt is being made to bring one of the DS3's spans in service but the DS3 is out of service
00 5B	DS3 Offset Out of Range	DS3 Offset is outside the valid range 0-27
00 5D	Invalid Data Value	Invalid data or data out of range
00 5E	DS3 In Service	An attempt is being made to loop back to a DS3, but the DS3 is in service
00 5F	Invalid Resource Attribute	
00 60	Outseize Failure, Guard Timing	Returned when the connection terminating party is in the guard state allowing for inseizures only.
00 61	Invalid Slot	The slot specified is not valid for the message sent.
00 62	Invalid Span Offset	Field reserved for span offset is outside the valid range for the line card (0-1 for 2-span, 0-3 for 4-span, 0-7 for 8-span).
00 63	Unassigned Span	The host has not yet assigned the specified Logical Span ID to a physical span.
00 64	Unassigned Span A	The host has not yet assigned that span number to a hardware location. If two span/channels exist in message, the first span address is unassigned.
00 65	Unassigned Span B	The host has not yet assigned that span number to a hardware location. If two span/channels exist in message, the second span address is unassigned.

00 66	Outseize Failure, Busied Out	Returned when connection terminating party is busied out.
00 67	Channel B Out of Service	Returned when connection terminating party is out of service due to a span alarm condition, is disabled, or both. Also returned if terminating party is going through error recovery or if no release of the distant end is detected.
00 68	Channel B Not Idle	Returned when the connection terminating party is currently involved in call setup or teardown with another call.
00 69	Span Already Assigned	Span selected to be assigned is already assigned.
00 6A	Span Offset Not Available	A span has been previously assigned to that slot and offset.
00 6B	Span Providing Loop Timing	Span selected to be unassigned has been selected as a loop timing source.
00 6C	Framing Error	An incorrect framing type was specified.
00 6D	Zero Suppression Error	An incorrect zero suppression type was specified.
00 6E	invalid Cable Length	An invalid cable length was specified.
00 6F	Card Out of Service	The card in the slot specified is not in service.
00 70	Single Matrix Out of Service Attempt	A single matrix cannot be taken out of service.
00 71	Channel Out of Service	One or more of the specified channels is out of service.
00 72	Span In Service	The span specified is in service.
00 73	Span Out of Service	The span specified is out of service.
00 74	Invalid Card Type	Incorrect card type for attempted operation.
00 75	Invalid Entity	The entity is out of range.
00 76	DSP SIMM in Service	The SIMM specified is in service.
00 77	Invalid DSP SIMM Value	The value specified for SIMM is invalid.
00 78	Invalid DSP Function Type	The type specified for SIMM is invalid.
00 79	Invalid DSP SIMM Configuration	The configuration specified for SIMM is invalid.
00 7A	Deletion of VRAS SIMM In Progress	An attempt has been made to download a recorded announcement to a SIMM while the SIMM is in the process of being deleted.
00 7B	Another Announcement Download is in Progress	An unsuccessful attempt has been made to download a recorded announcement while another recorded announcement download is in progress.
00 7C	No Recorded Announcement Download Initiate Message Received	A <i>Recorded Announcement Download</i> message has been sent without a prior <i>Recorded Announcement Download Initiate</i> message sent.
00 7D	CCS Redundant I/O Card Not Available	ISDN or SS7 card in a redundant system is missing or unavailable.
00 7E	RAN Download to Out of Service DSP	An attempt was made to download a recorded announcement to a DSP that is out of service.
00 7F	Software module still locked.	This value is returned when attempting to configure a locked module for which no product license has been downloaded.
00 80	Partial Dial Condition	The receive inter-digit timer has expired.
00 81	Permanent Signal Condition	The 1st receive digit timer has expired.
00 82	Failure to Receive Wink 1	The maximum Receive Wink 1 detection timer has expired.
00 83	Failure to Receive Wink 2	The maximum Receive Wink 2 detection timer has expired.
00 84	Failure to Receive Wink 3	The maximum Receive Wink 3 detection timer has expired.
00 85	Failure to Receive Wink 4	The maximum Receive Wink 4 detection timer has expired.
00 86	Failure to Receive Wink 5	The maximum Receive Wink 5 detection timer has expired.
00 87	Failure to Receive Wink 6	The maximum Receive Wink 6 detection timer has expired.
00 88	Failure to Receive Wink 7	The maximum Receive Wink 7 detection timer has expired.
00 89	Failure to Receive Wink 8	The maximum Receive Wink 8 detection timer has expired.

00 8A	No SIMM In Service For Requested Function	There is no SIMM currently in service and configured for tone generation or VRAS.
00 8B	No Outpulsing Data Available	Outpulse data is not available for the stage indicated. This response is also returned when connecting to an idle SS7 channel.
00 8C	No Action ICBs in Message	An <i>Outseize Control</i> or <i>Inseize Control</i> message was sent without any action ICBs included.
00 8D	No Data ICBs in Message	An <i>Outseize Control</i> message was sent without a required data ICB included.
00 8E	Wink Tolerance Exceeded	Given incoming wink exceeded maximum allowed duration.
00 8F	Failure to Detect Off-hook	An ANI request off-hook was not detected within the detection interval.
00 90	Failure to Detect Dialtone	Dialtone not detected within detection interval.
00 91	Inpulsed Stage Not Collected	Collection of given stage of incoming digits not indicated.
00 92	Inpulsing Complete Timeout	Time for complete incoming digit string collection for a given stage exceeded maximum time allowed.
00 93	Invalid Inpulsed Source Span/Channel	An invalid span/channel is indicated as the inpulsing channel.
00 94	No Valid Inpulsed Data to Outpulse	An Outpulse Stage N with Previously Inpulsed Digits instruction was initiated, however, the specified inpulsing has not occurred or has failed.
00 96	Invalid Configuration Request for Span	Specified configuration request for span is invalid.
00 98	No ULC Module	ULC module is missing.
00 99	Invalid Configuration Request	Configuration request is invalid.
00 9A	Invalid Command for Card Type, Span, or Channel	Message does not apply to this card, span, or channel.
00 9B	Illegal D Channel Function	For ISDN D channel call control administration only.
00 9C	Invalid ISDN D Channel	ISDN PRI D channel is not valid.
00 9D	Invalid ISDN B Channel	ISDN PRI B channel is not valid.
00 9E	Invalid ISDN Facility	ISDN facility is invalid.
00 9F	Invalid Value for Entity	The value specified for entity is invalid.
00 A0	SIMM Not Present	SIMM specified is not present.
00 A1	D Channel Assigned	A D channel is assigned on span ID selected.
00 A2	Called Party Mandatory for ISDN Setup	The called party digits are mandatory instruction elements over ISDN PRI.
00 A3	Card Not Ready for Configuration	The card is not in a valid state to accept configuration data.
00 A4	ISDN D Channel Exceeds Maximum	The ISDN card specified can not be configured for another D channel. Check the card's address mode.
00 A5	Invalid Option for Protocol	Protocol assigned to channel in ranges does not allow for the function the host is trying to perform.
00 A6	PPL Table Size Exceeds Max	Size of table host wishes to download is too large.
00 A7	PPL Table Already Exists	Table ID already in use.
00 A8	PPL Table Does Not Exist	Table ID referenced does not yet exist.
00 A9	PPL Table Does Not Validate	The table the host has downloaded does not validate. Further information can be found in Byte and Entity fields of message responding to this error.
00 AA	Insufficient Hardware	The function the host is trying to perform can not be performed because the hardware needed is not present.
00 AB	PPL Table Part of Protocol	Error occurred trying to delete a Primitive Table or State/Event Table. The protocol that the table is associated with must be deleted first.
00 AC	Invalid Command for ISDN Facility	The command for the ISDN facility was not valid.

00 AD	DSP Resource Inconsistency	Returned in a <i>Collect Digit String</i> or <i>DSP Service Request</i> message if digit collection is already active on the specified channel. Also returned when the host attempts to configure two transmitters on one card at the same time.
00 AE	Hardware Not Configured for DSP Function	The DSP resource management software has been requested to provide a DSP resource which does not exist. Please check DSP configurations of DSP card by sending a <i>Card Status Query</i> message.
00 AF	PPL Layer Forward Signal Timeout	Forward R2 Signal has not been detected within the maximum amount of time allowed.
00 B0	PPL Layer Backward Signal Timeout	Backward R2 Signal has not been detected within the maximum amount of time allowed.
00 B1	PPL Layer Premature Answer	Answer was detected on an outgoing call before call setup completion.
00 B2	PPL Layer Congestion Received	Backward R2 Congestion Signal has been detected.
00 B3	System Table in Use	System software is using the table you are trying to modify/create. Please wait a few seconds and try again.
00 B4	Line Card Response To Matrix Timeout	The matrix timed-out waiting for a response from a line card. Check configuration for channel ranges that span more than two cards.
00 B5	CPA Member Exceeds Maximum For Class	The maximum number of call progress analysis patterns which can be included in a class has been reached.
00 B6	CPA Pattern Exceeds Maximum	The maximum amount of call progress analysis patterns supported by the system has been reached.
00 B7	CPA Tone Group Exceeds Maximum	The maximum amount of call progress analysis tone groups supported by the system has been reached.
00 B8	Invalid Pattern ID	An invalid pattern ID has been specified for call progress analysis.
00 B9	Invalid Interval Descriptor Count	An invalid interval descriptor count has been specified for call progress analysis.
00 BA	CPA Classes Exceeds Maximum	The maximum amount of call progress analysis classes supported by the system has been reached.
00 BB	Invalid CPA Class	An invalid call progress analysis class was specified in a configuration message.
00 BC	Invalid Data Found at Byte N	The message became invalid because of data found at byte offset N. See the LSB of the <i>Status</i> field of the message for the byte offset where data became invalid.
00 BD	Invalid Table Type	The table type specified is invalid.
00 BE	Invalid Table ID	The table ID specified is invalid.
00 BF	ISDN Response Wait Timeout	ISDN-initiated message timed-out; waiting for a response from the network (for example, Register with no Release Complete).
00 C0	Action Denied	Action request denied.
00 C1	Action Rejected	Action request rejected.
00 C2	Slot Assigned As Standby	The slot number is assigned as a standby slot.
00 C3	Slot Not Assigned As Standby	The slot number is not assigned as a standby slot.
00 C4	Standby I/O Relay State Mismatch	The states of the relays of the I/O cards specified do not match (relays on one are enabled; relays on the other are disabled).
00 C5	Incompatible Line Card Types	The line cards specified are not compatible.
00 C6	Redundant I/O Cards Not Present	Redundant I/O cards are not present in one or more of the slots specified.
00 C7	Line Card Switchover Already Completed	The <i>Line Card Switchover</i> requested has already been completed.
00 C8	Invalid SIMM Type	Trying to perform VRAS functions on a non-VRAS SIMM.

00 C9	Invalid Recorded Announcement ID	The recorded announcement ID is not in the allowable range or the host has requested a recorded announcement ID that does not exist in the system. <u>Allowed Range</u> MFDSP/DSP-ONE: 0 - 4,095 DSP Series 2: Not Applicable
00 CA	Recorded Announcement Already Exists	The host has attempted to download an announcement using a recorded announcement ID that is already stored in the system.
00 CB	Insufficient Memory on VRAS SIMM	Not enough memory to store the specified recorded announcement.
00 CC	Maximum Recorded Announcement Limit Reached on VRAS SIMM	The host has attempted to download more than the maximum allowed recorded announcements to a particular VRAS SIMM. MFDSP - 300 DSP-ONE - 2,048 DSP Series 2 - Does Not Apply
00 CD	Invalid IE Length	The host has sent an information element (IE) in an ICB with an incorrect length.
00 CE	Invalid IE Count	The host has sent an ICB with an incorrect <i>IE Count</i> value.
00 CF	Invalid Component Type	The host has sent an ICB with an incorrect component type.
00 D0	ISDN D Channel Switchover Timed Out	The D channel switchover could not be completed due to timer expiration.
00 D1	Invalid ISDN Connection Endpoint Identifier	A valid Connection Endpoint Identifier could not be found for the addressed D channel.
00 D2	Invalid AIB	The address information block (AIB) is not in the proper format, or it addresses an invalid PPL state machine or channel.
00 D3	Invalid Protocol ID	A protocol ID specified in a PPL message is invalid.
00 D5	Channel Already In Service	Applies to ISDN and SS7 only. Indicates that you attempted to either assign or de-assign a protocol to a channel that is in service.
00 D6	Invalid Command For Trunk Front End	An invalid command was sent from the host for a trunk front end.
00 D7	End Of PPL Audit Data	The end of the PPL auditing data has been reached. There are no more PPL auditing blocks in the PPL auditing logs.
00 D8	Invalid SS7 Signaling Stack ID (0-3)	A signaling stack was specified that either: Was greater than the allowed values (0-3), or Did not match the signaling stack for the primary resource
00 D9	SS7 Signaling Stack Already Configured	An attempt was made to configure an SS7 signaling stack that is already configured.
00 DA	Invalid SS7 Module Count	An attempt was made to configure an SS7 signaling stack with an invalid number of modules. The module count must be 3.
00 DB	Duplicated SS7 Module	An SS7 signaling stack configuration attempt specified the same SS7 model more than once.
00 DC	Unsupported SS7 Module Variant	An SS7 signaling stack configuration attempt specified an unsupported SS7 module variant.
00 DE	Invalid SS7 Module Type	An SS7 signaling stack configuration attempt specified an invalid SS7 module type.
00 DF	Invalid SS7 Signaling Stack Configuration	An SS7 signaling stack configuration attempt could not be interpreted correctly.
00 E0	SS7 Signaling Stack Not Configured	An SS7 signaling stack configuration attempt specified a signaling stack ID that is not configured.
00 E1	Invalid SS7 Signaling Link ID	An invalid SS7 signaling link ID was specified that was either: Too large for the SS7 card model, or An attempt was made to de-configure a link that is not configured.

00 E2	Invalid SS7 Signaling Link Set ID	An invalid SS7 signaling link set ID was specified. Valid Link Set IDs are 0 to 35.
00 E3	SS7 Signaling Link Set Not Configured	An SS7 configuration attempt specified a signaling link set that is not configured.
00 E4	SS7 Signaling Link Already Configured	An SS7 configuration attempt expected the specified SS7 signaling link either: Was not configured, or Did not have configured SS7 signaling links
00 E5	Invalid SS7 SLC	An SS7 configuration attempt specified an invalid SLC. Valid SLC codes are 0 to 15.
00 E6	SS7 Signaling Link Set Already Configured	An SS7 configuration attempt expected either: The specified SS7 signaling set was not configured, or The specified SS7 signaling stack had no configured SS7 signaling link sets
00 E7	SS7 APC Already Configured	An SS7 configuration attempt specified an adjacent point code (APC) that has already been associated with another link set. All links within a SS7 signaling stack that link to an adjacent signalling point must belong to a single link set.
00 E8	Invalid SS7 Signaling Link Data Rate	An SS7 configuration attempt specified an invalid signaling link data rate. Valid data rates are 64 Kbps (0) and 56 Kbps (1).
00 E9	SS7 SLC Already Configured	An SS7 configuration attempt specified an SLC that has already been assigned to another link within the same link set.
00 EA	Invalid DCE-DTE Configuration	An SS7 configuration attempt specified an invalid DCE-DTE configuration.
00 EB	SS7 Signaling Route Already Configured	One of the following is true: An SS7 signaling route configuration specified a route ID that already exists, or An SS7 signaling route configuration attempted to de-configure a destination for which CICs are still configured. An SS7 signaling linkset configuration attempted to de-configure a linkset for which routes are still configured.
00 EC	Invalid SS7 Signaling Route Mode Configuration	An <i>SS7 Signaling Route Configure</i> message failed due to one of the following: It specified a previously configured destination ID, but the specified stack ID or DPC did not match the previously configured destination ID. It used a destination ID that was not previously configured, but a previously configured destination ID for the signaling stack has a matching DPC. It reused the same link set to the same DPC (Route duplication) An <i>SS7 Signaling Route Query</i> message failed due to one of the following: An invalid destination ID (valid IDs are 0 to 63) An invalid route ID (valid IDs are 0 to 254) Either the specified route ID or destination ID is not configured to the specified stack ID The specified destination ID and route ID are inconsistent
00 ED	No SS7 Signaling Route Entry Available	An <i>SS7 Signaling Route Configure</i> message was received and there were no unused route table entries. The CSP supports up to 20 routes per stack.
00 EE	SS7 Signaling Destination Not Configured	An SS7 configuration attempt specified a destination in the SS7 signaling stack that was not previously configured.

00 EF	No SS7 Signaling DPC Entry Available	An <i>SS7 DPC-CIC Configure</i> message was received specifying a DPC when no more DPC table space is available. The system allows for one DPC per span. If you try to configure more than 128 CIC groups on one system, you will get this message.
00 F0	CCS Redundancy Already Configured	A CCS redundancy configuration attempt specified a primary and/or secondary slot that has previously been configured.
00 F1	CCS Redundancy Not Configured	A CCS redundancy configuration or query message was received before a <i>CCS Redundancy Configure</i> message was received.
00 F2	CCS Redundancy: Not A Primary Slot	The <i>CCS Redundancy Configure</i> message attempted to reference the secondary card in a CCS redundant pair. You must reference the primary card only.
00 F3	Channel Already an SS7 CIC	An <i>SS7 CIC Configure</i> message was received that specified a CIC (or group thereof) that has been previously configured for the same channel.
00 F4	Primary SS7 Slot Same As Secondary Slot	An <i>CCS Redundancy Configure</i> message was received specifying the same slot as the primary and secondary slot.
00 F5	No SS7 Signaling Links Configured	An SS7 configuration attempt specified one of the following: An SS7 signaling link that is not configured An SS7 signaling link set that has no configured SS7 signaling links
00 F6	No SS7 Signaling Routes Configured	An SS7 configuration attempt specified a destination ID or route ID that was not configured.
00 F7	SS7 DPC-CIC Already Configured	An <i>SS7 DPC-CIC Configure</i> message was received that specified a DPC-CIC (or group thereof) that has been previously configured for the same signaling stack.
00 F8	No SS7 Configuration Entity Available	An SS7 signaling route configuration attempt failed because it specified an entity (destination ID or route ID) that was not currently configured, but may have been configured previously. An SS7 DPC-CIC configuration attempt failed because no free DPC-CIC resources were available.
00 F9	Invalid Number of SS7 ISUP Parameters	An <i>SS7 ISUP Message Configure</i> message was sent with an invalid number of parameters.
00 FA	Invalid SS7 ISUP Message Type	An <i>SS7 ISUP Message Configure</i> message was sent with an illegal Message Configuration Entry value.
00 FB	Node Already Exists	The <i>Assign Logical Node ID</i> message failed because node is already assigned.
00 FC	Invalid Node ID	A node ID is invalid because it met one of the following conditions: Specified node ID is greater than maximum value of 31 (0x1F). Specified node ID is greater than maximum value allowed when CSP conferencing is enabled.
00 FD	Node ID Not Configured	Assignment of spans failed because node is not yet assigned. When sending <i>SS7 CIC Configure</i> (0x6A) over CSP, this response value indicates the Ethernet is not available to the SS7 I/O card and you should check the connection.
00 FE	Maximum Number of Redundant Objects Reached	
11 11	Invalid Module	Module specified is invalid (Valid range is 0x00 - 0x03)
11 12	Module Not Present	Module specified is not present
11 13	Module Not Ready	Module is not in a state to receive messages
12 00	SIP Common Error	
12 01	Invalid IP Signaling Series 3 Card ID	Invalid IP Signaling Series 3 Card ID used, or ID is out of the valid range
	User Not Found	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
12 02	Invalid TLV tag	

12 03	MatrixInternal error	
	Invalid PPL event	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
12 04	Configuration Not Completed	
	Invalid Data	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
12 05	Invalid Query Type	
12 06	IP Signaling Series 3Card ID already configured	
12 07	Invalid Type	
12 08	Invalid State	Trying to set an invalid service state or the Matrixis already in this state For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
12 09	Invalid IPDC Usage option	Trying to set an invalid IPDC usage or the option is out of range
12 0A	Invalid IP Signaling Series 3 Card interface support	Trying to set an invalid IP Signaling Series 3 Card interface support
12 0B	Invalid Base Value	Trying to set an invalid base value for channels and spans
12 0C	IP Signaling Series 3 Card ID not configured	Response to Query
12 0D	IP Signaling Series 3 Card not in service	Response to Query
13 00	SIP Common Error	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
13 01	User Not Found	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
13 02	Invalid Call Handle	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
13 03	Invalid PPL Event	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
13 04	Invalid Data	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
13 05	Network Error	Remote end rejected the message. For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
13 06	Timer Expired	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
13 07	No more PPL state machines available	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
13 08	Invalid State	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
13 09	Invalid Time Slot	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
13 0A	VDAC Rejected	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
13 0B	Authentication Failed	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages.
13 0C	Call not in bearer free mode	
13 0D	Invalid Action Value	For SIP, used in <i>Route Control</i> and <i>PPL Event Request</i> messages. In <i>PPL Event Request</i> indicates the socket is not open so it cannot be closed.
13 12	DNS Server Disabled (NACK)	All <i>Route/Outseize Control</i> API messages are nacked with this response if the outbound SIP call requires a DNS Server lookup and the DNS is disabled.
14 01	Invalid Config Object Type	
14 02	Invalid or Out-of-Range Config Instance ID for the IP Signaling Series 3 Card	
14 03	Invalid Query Type	
14 04	Invalid TLV tag	
14 05	IP Signaling Series 3 Card Internal Error	
14 06	Invalid TLV value	
14 07	Invalid ICB	
14 08	Configuration was not completed	
14 09	IP Signaling Series 3 Card ID already created	
14 0A	Invalid State	Trying to set an invalid service state or the call server is already in this state

14 0B	CSP Matrix Series 3 Card ID already created	
14 0C	Invalid CSP Matrix Series 3 Card ID	
14 0D	Invalid Slot Number	
14 0E	IP Signaling Series 3 Card is not configured	
15 01	Gateway mode not supported over ring	Connecting gateway mode VoIP channel to B timeslot over ring is not supported
17 01	Invalid DSP Number	Used in messages <i>Service State Configure</i> , <i>DSP SIMM Configure</i>
17 02	File ID Not Found	Used in message <i>DSP Cache Modify</i> .
17 03	No Active Playback/Record Session	Used in messages <i>Play File Modify</i> , <i>Play File Stop</i> , <i>Record File Modify</i> , <i>Record File Stop</i>
17 04	Playback/Record Session in Incorrect State	Used in messages <i>Play File Modify</i> , <i>Record File Modify</i>
17 05	Invalid File ID for Configuration	Used in messages <i>Play File Start</i> , <i>Record File Start</i>
17 07	No Resources Due to DSP Overload	Used in messages <i>Resource Create</i> , <i>Play File Start</i> (to channel or conference), <i>Record File Start</i> (to channel or conference), <i>Conference Create</i> , <i>DSP Service Request</i> , <i>Collect Digit String</i> , <i>Recorded Announcement Connect</i> , <i>Connect Tone Pattern</i> , <i>Outpulse Digits</i>
17 0A	Unable to remove party from a conference	Used in <i>Resource Connect</i> and <i>Resource Disconnect</i> messages
17 0B	Unable to add party to a conference	Used in <i>Resource Connect</i> and <i>Resource Disconnect</i> messages.
17 0C	Channel not present in parent conference	Used in <i>Resource Connect</i> message
17 0D	Channel not present in child conference	Used in <i>Resource Disconnect</i> message
17 0E	Reached System Maximum Number of Conferences	Used in <i>Resource Create</i> and <i>Conference Create</i> messages.
17 11	Not supported in gateway mode	Used in messages <i>Connect to Conference</i> , <i>Resource Connect</i>

MSB 0x18 - Invalid B Channel State

The LSB indicates the current call state.

If the value is 0x1850 or above, it indicates the Layer 3 State. The Layer 3 state values depend on the signaling protocol. See the appropriate *.ppl file (CD ROM only) for Layer 3 state values, or contact Dialogic technical support.

If the value is lower than 0x1850, it indicates the Layer 4 State. The Layer 4 states are defined below. Some values are not backward-compatible with System Software versions earlier than 5.3.

18 00	Out of Service
18 01	Incoming Call
18 02	Wait For CSA in Incoming Call State
18 03	In Service Idle
18 04	Incoming Call, Wait for Host Connect
18 05	Outgoing Call, L3 Outseize Wait
18 06	Wait For CSA in Incoming Call Wait for Host State
18 07	Answered/Connected
18 08	L4 Clear ACK Wait, L3 Disconnect Received
18 09	L3 Clear Wait
18 0A	Busied out
18 0B	Outgoing Call Cut-thru
18 0C	Wait for CSA in Incoming Alerted State
18 0D	Wait for CSA in Incoming in Answered State
18 0E	Incoming Call Alerted
18 0F	L3 Clear Wait
18 10	Wait for CSA in Outgoing Cut-thru
18 11	Wait for CSA in Outgoing Alerted State

18 12	L4 Clear ACK Wait, L3 Clear Received
18 13	Outgoing Call Alerted
18 14	L4 Clear ACK Wait, L5 Clear Received
18 15	L5 Release Wait
18 16	Incoming Call, L4 Clear ACK Wait (Park Processing)
18 17	Incoming Call Alerted, L4 Clear ACK Wait (Park Processing)
18 18	Answered, L4 Clear ACK Wait (Park Processing)
18 19	Outgoing Alerted, L4 Clear ACK Wait (Park Processing)
18 1A	Outgoing Cut-thru, L4 Clear ACK Wait (Park Processing)
18 1B	Wait for CSA for Internal Routing
<p>MSB 0x1D - Invalid A Channel State The LSB indicates the current call state. If the value is 0x1D50 or above, it indicates the Layer 3 State. The Layer 3 state values depend on the signaling protocol. See the appropriate *.ppl file (CD ROM only) for Layer 3 state values, or contact Dialogic technical support.</p> <p>If the value is lower than 0x1D50, it indicates the Layer 4 State. The Layer 4 states are defined below. Some values are not backward-compatible with System Software versions earlier than 5.3.</p>	
1D 00	Out of Service
1D 01	Incoming Call
1D 02	Wait For CSA in Incoming Call State
1D 03	In Service Idle
1D 04	Incoming Call, Wait for Host Connect
1D 05	Outgoing Call, Layer 3 Outseize Wait For PPL Event Request message (0x0044): Outseize Acknowledgment Network Wait
1D 06	Wait For CSA in Incoming Call Wait for Host State
1D 07	Answered/ Connected
1D 08	L4 Clear ACK Wait, Layer 3 Disconnect Received For PPL Event Request message (0x0044): Layer 4 Release Wait
1D 09	Layer 3 Clear Wait For PPL Event Request message (0x0044): Network Release Wait
1D 0A	Busied out
1D 0B	Outgoing Call Cut-thru For PPL Event Request message (0x0044): Externally Outseized
1D 0C	Wait for CSA in Incoming Alerted State For PPL Event Request message (0x0044): Hold Acknowledge Wait
1D 0D	Wait for CSA in Incoming in Answered State For PPL Event Request message (0x0044): Layer 3 Answer Wait
1D 0E	Incoming Call Alerted For PPL Event Request message (0x0044): Layer 4 Answer Wait
1D 0F	L3 Clear Wait
1D 10	Wait for CSA in Outgoing Cut-thru For PPL Event Request message (0x0044): Layer 4 Recall Wait
1D 11	Wait for CSA in Outgoing Alerted State For PPL Event Request message (0x0044): Purge Response Wait
1D 12	L4 Clear ACK Wait, L3 Clear Received For PPL Event Request message (0x0044): Purge Wait

1D 13	Outgoing Call Alerted For PPL Event Request message (0x0044): Park	
1D 14	L4 Clear ACK Wait, L5 Clear Received	
1D 15	L5 Release Wait	
1D 16	Incoming Call, L4 Clear ACK Wait (Park Processing)	
1D 17	Incoming Call Alerted, L4 Clear ACK Wait (Park Processing)	
1D 18	Answered, L4 Clear ACK Wait (Park Processing)	
1D 19	Outgoing Alerted, L4 Clear ACK Wait (Park Processing)	
1D 1A	Outgoing Cut-thru, L4 Clear ACK Wait (Park Processing)	
1D 1B	Wait for CSA for Internal Routing	
48 01	Invalid Number of TLVs	The number of TLVs is out of range.
48 02	Invalid Module Number	The module number is invalid for the message sent.
48 03	Invalid IP Address	The IP address out of range or an invalid class.
48 04	Invalid Subnet Mask	The subnet mask is out of range or is invalid for the IP address.
48 05	Duplicate IP Address	The IP address is already assigned to another module number.
48 06	Invalid TLV tag	The TLV tag is out of range.
48 07	Invalid Data Type	The Data Type is out of range.
48 08	Invalid TLV combination	The combination of TLVs is invalid.
48 09	Module Not Present	
48 11	Socket Incorporate Operation Error	
48 12	Invalid Parameter Length	
48 0A	Invalid IMMSG	
48 0B	Motherboard IP Address Not Configured	
48 0C	Invalid Ethernet Port	
48 0D	Invalid TLV Value	
48 0F	Socket bind to user error	
4D 01	Software key format not valid.	The first two bytes of the Product License field are not defined key types, or the product license contains invalid data.
4D 02	Data decrypted not valid.	A serial number decrypted from a product license cannot be matched against an existing serial number on a CSP.
4D 03	Product License - Insufficient Licensed Resources	
4D 3D	Product License - Invalid ICB Data	
4D 74	Invalid Card Type.	The line card that license was generated for does not support licensing
4D AA	Insufficient Hardware.	Not enough room for the number of licensed spans on the chassis
50 01	Response Timeout	
50 02	Null Buffer	
50 03	Queue Full	
50 04	Feature Disabled	
50 05	No ICBs	
50 06	Invalid ICB Type (Message Not Processed)	
50 07	Invalid ICB Type (Message Partially Processed)	
50 08	Invalid ICB Subtype (Message Not Processed)	
50 09	Invalid ICB Subtype (Message Partially Processed)	
50 0A	Invalid Message (Message Not Processed)	
50 0B	Invalid Message (Message Partially Processed)	
50 0C	Unauthorized Command	

50 0D	Incorrect ICB Data Length	
50 0E	Host Not Connected	
50 0F	Incorrect Host ID	
50 10	Reserved	
50 11	Maximum Number of Queries Exceeded	
50 12	Not used	
50 13	Not used	
50 14	Multi Host Download In Progress	
50 15	Multi Host Being Enabled, Wait 32 Sec In the SS7 CLLI Query message (0x00EB): The message could not be processed for either of the following reasons: - Stack ID and CIC Group mismatch - Internal error	
50 16	Odd ICB Data Count (Extended API)	
50 17	Non-Unique Host ID In the SS7 CLLI Query message (0x00EB): Invalid configuration data format, if the data format specified is not 0x01 (TLV format)	
50 18	Invalid ICB Subtype	
50 19	Host ID Assigned to Different Host	
50 1A	Invalid ICB Data Entry	
50 47	Invalid Ring Loop Timing.	Trying to set a node to be master-configurable, but the ring timing has already been derived, or trying to set ring timing on a master node or master- configurable node, or trying to set secondary timing on the EXNET-ONE card.
50 AA	Insufficient Hardware.	This response status is returned when trying to initiate the Enhanced Fault Tolerance Ring Mode, but the EXNET-ONE and I/O are installed. (These EXNET® cards cannot be used in the Enhanced Fault Tolerance Ring Mode).
50 C0	Action Denied.	A value of Action Denied (0xC0) is returned if the user is trying to set an EXNET-ONE card.
50 75	Invalid TLV Tag.	The TLV specified is not one of the 4 types allowed (Trunk Number, Local CLLI, Remote CLLI, Trunk Type)
50 CD	Invalid TLV Tag Length	
50 E0	SS7 Signaling Stack Not Configured	
50 F9	Invalid TLV Count. The TLV Count should be 0x04.	
51 00	ISUP Congestion	Local or remote ISUP congestion scenario. Only priority calls will be allowed in congestion state.
51 01	End of Route Table	
51 02	Retry Limit Exceeded	
51 03	Termination Error	
51 04	Purge	
51 05	No Destination Found	
51 06	Recursion	
51 07	ACK Wait Timer Expired	
51 08	No Common Resource Group Found	
51 0A	Termination Validation Error	
51 0B	Channel unavailable due to initialization or OOS	
51 0C	End of Entry Data	

51 0D	Invalid Routing Method	
51 0E	Invalid Resource Group Table	
51 0F	Invalid Route Table	
51 10	Reserved	
51 11	Invalid Resource Group	
51 12	Invalid Criteria	
51 13	Invalid Entry Data	No available routes
51 14	System Error	
51 15	Illegal Instruction	
51 1E	V5 ID Invalid	
51 1F	V5 User Port Invalid	
51 20	V5 User Port In Use	
51 21	V5 User Port Blocked	
51 22	V5 TS Alloc Failed	
51 23	V5 Timer Expiry	
51 24	V5 L3 Internal Error	
51 25	V5 Formie Invalid	
51 26	V5 Invalid Num of Formies	
51 28	V5 Unspecified	
51 29	V5 Access NW Fault	
51 2A	V5 Access NW Blocked	
51 2B	V5 Connection Present at PSTN up to a Diff V5TS	
51 2C	V5 Connection Present at V5TS to a Diff ISDN UPTS	
51 2D	V5 Connection Present at ISDN up to a Diff V5TS	
51 2F	V5 Dealloc Cannot Comp Incompatiable Data	
51 30	V5 Dealloc Cannot Comp V5TS Data Incompatible	
51 31	V5 Dealloc Cannot Comp Port Data Incompatible	
51 32	V5 Dealloc Cannot Comp Upts Data Incompatible	
51 33	V5 User Port Not Provisioned	
51 34	V5 TS Invalid Identified	
51 35	V5 Link Invalid Identified	
51 37	V5 TS Used as Physical C Channel	
51 38	V5 Link Blocked	
51 8F	Gatekeeper routed but no gatekeeper address	
51 90	H245 end session received	
51 91	RAS DRQ	
51 92	CALL PROCEEDING has protocol error	
51 93	H225 timer expired	
51 94	REL COMPLETE received	

51 95	Incoming H225 message has Q931 protocol error	
51 96	No Remote Alias Address or SRC IP/Port not Available	
51 97	H.323 Protocol Error (H.225, H.245, or RAS)	
51 98	H.225, H.245 Socket Open Fail	
51 99	Gatekeeper not Configured	
51 9A	LRQ Failed	
51 9B	ARQ Failed	
51 9C	No Associated Timeslot	
51 9D	DSP Resource Wait Timeout	
51 9E	Invalid DSP Resource Request	Resource already attached.
51 20	V5 ID Invalid	
51 21	V5 User Port Invalid	
51 22	V5 User Port Blocked	
51 23	V5 Timer Expiry	
51 24	Unexpected Clearing Received	
51 25	Invalid V5 Formatted IEs	
51 26	Invalid Number of V5 Formatted IEs	
52 01	Signaling Route Test (SRT) failed.	
52 02	Signaling Route Test Acknowledgment (SRTA) has an improper pattern.	
52 03	The SRTA was received on the wrong SLC. (This error code not supported for DDI).	
52 04	Link congested	
52 05	Link out of service	
52 06	Link not equipped	
52 07	Request was terminated by the Host	
52 08	SRT is already running	
52 09	System error	
52 0A	SRTC timed out	
52 0B	Received valid SRTA in wrong Link ID (This error code not supported for DDI)	
52 0C	Route not configured for this DPC	
52 81	Invalid module type for DPC/Stack/CIC combination. (Also used in the <i>PPL Timer Configure</i> message: 0x00CF)	
52 82	Message received for unassigned CIC.	
52 83	Only Extended API is supported for VCIC.	
If the MSB is 0x54, Error Detected in SCCP/TCAP, the LSB indicates the error as follows:		
54 00	Error Unknown	
54 01	Incorrect ICB	The maximum number of ICBs for the TCAP primitive set is 16.
54 02	Incorrect ICB Length or Parameter	
54 03	Incorrect or Missing Mandatory TCAP Parameters	
54 04	Incorrect or Missing Mandatory SCCP Parameter	
54 05	Invalid Primitive (Primitive Not Supported)	
54 06	Instance does not Exist	
54 07	Primitive Not Expected as this Time	
54 08	Mandatory Component Missing	
54 09	TCAP simultaneous active dialogue/ transaction/operations reached the maximum limit.	SS7 system busy condition (for CPU, memory or MCB).

54 0A	SSN is not Configured or not Active
54 0B	Maximum Length of TCAP Components for a Message Exceeded
55 00	<i>PPL Event Request: Error Unknown</i> <i>SCCPTCAP Configure: No More Status Info, or Status Unknown</i>
55 01	<i>PPL Event Request: Invalid ICB or a Mandatory ICB is missing</i> <i>SCCPTCAP Configure: Invalid Config Type</i>
55 02	<i>PPL Event Request: Incorrect ICB Length or Parameter</i> <i>SCCPTCAP Configure: Subsystem Configured</i>
55 03	<i>PPL Event Request: Incorrect or Missing Mandatory TCAP Parameter</i> <i>SCCPTCAP Configure: Subsystem Not Configured</i>
55 04	<i>PPL Event Request: Incorrect or Missing Mandatory SCCP Parameter</i> <i>SCCPTCAP Configure: Subsystem Allowed</i>
55 05	<i>PPL Event Request: Invalid Primitive (Primitive Not Supported)</i> <i>SCCPTCAP Configure: Subsystem Not Allowed</i>
55 06	<i>PPL Event Request: Instance does not Exist</i> <i>SCCPTCAP Configure: Invalid Option</i>
55 07	<i>PPL Event Request: Primitive Not Expected at this Time</i> <i>SCCPTCAP Configure: Exceeds minimum number of hosts for a Subsystem</i>
55 08	<i>PPL Event Request: Mandatory Component Missing</i> <i>SCCPTCAP Configure: Invalid Parameter</i>
55 09	<i>PPL Event Request: Resource Limitation</i> <i>SCCPTCAP Configure: Invalid Parameter Length</i>
55 0A	<i>PPL Event Request: SSN is not Configured or not Active</i> <i>SCCPTCAP Configure: Parameter Configured</i>
55 0B	<i>PPL Event Request: Maximum Length of TCAP Components for a Message Exceeded</i> <i>SCCPTCAP Configure: Parameter Not Configured</i>
55 0C	Invalid SCCP Upper Layer
55 0D	Maximum Active Subsystems Exceeded
55 0E	DPC Not Configured

6 PPL Component Information

Overview

Purpose This chapter contains information regarding PPL Components for PPL software development on the CSP.

PPL Component IDs

This section lists the PPL Component IDs used in PPL software development for the CSP. The component IDs are listed by protocol.

Important! All the PPL components listed in this section support the extended PPL table type except for those showing otherwise.

Component Modification	Components that are released for modification using the PPL Composer are marked with an asterisk (*).
T1	0x0003 T1 PPL Component* (Supports basic PPL table type only)
E1	0x0001 E1 PPL Component* (Supports basic PPL table type only)
Layer 4/Layer 5 Call Control PPL Information	0x0061 Channel Management*
	0x0062 Call Management*
	0x0063 Physical Connection Management*
	0x0064 Router*
	0x0084 Interworking
	0x0086 MCC Group
	0x0087 MCC Channel
SS7	L3P
	0x000F L3P CIC (CIC Call Control)*
	0x0010 L3P Link (Link Management)*

ISUP

0x0012 ISUP CPC (Call Processing Control) *

0x0013 ISUP SPRC (Signaling Procedure Control)*

0x0014 ISUP CCI (Continuity Check Incoming)

0x0015 ISUP CRI (Continuity Recheck Incoming)

0x0016 ISUP BLS (Blocking/Unblocking Signal Sending)

0x0017 ISUP BLR (Blocking/Unblocking Signal Reception)

0x0018 ISUP CRS (Circuit Reset Sending)

0x0019 ISUP CRR (Circuit Reset Reception)

0x001A ISUP UCIC (Unequipped CIC Reception)

0x001B ISUP CGRS (Circuit Group Reset Sending)

0x001C ISUP CGRR (Circuit Group Reset Reception)

0x001D ISUP GBUS (Circuit Group Blocking/Unblocking Sending)

0x001E ISUP GBUR (Circuit Group Blocking/Unblocking Reception)

0x001F ISUP HGBR (Hardware Oriented Circuit Group Blocking/
Unblocking Reception)

0x0042 ISUP HLB (Hardware Failure Oriented Locally Blocking State)

0x0043 ISUP HRB (Hardware Failure Oriented Remotely Blocking)

0x0044 ISUP CRCR (Continuity Recheck Reception)

0x0045 ISUP MGBS (Maintenance Oriented Circuit Group Blocking/
Unblocking Sending)

0x0046 ISUP MGBR (Maintenance Oriented Circuit Group Blocking/
Unblocking Reception)

0x0047 ISUP HGBS (Hardware Oriented Circuit Group Blocking/
Unblocking Sending)

0x0076 ISUP CQS(Circuit Query Sending)

0x0077 ISUP CQR (Circuit Query Receiving)

0x0078 ISUP CVS (Circuit Validation Sending)

0x0079 ISUP CVR (Circuit Validation Receiving)

0x0080 ISUP CCO (Continuity Check Outgoing)

0x0081 ISUP CRCS (Continuity Check Resending)

0x0082 ISUP DCO (Demand Continuity Outgoing)

0x0083 ISUP CRO (Continuity Recheck Outgoing)

0x0085 ISUP SSC (Simple Segmentation Control)

0x00A8 ISUP ACC (Automatic Congestion Control)

TUP

0x0011 L3P TUP (Telephone User Part)*
0x0052 TUP CPC (Call Processing Control)*
0x0053 TUP SPRC (Signaling Procedure Control)*
0x0054 TUP BLR (Blocking/Unblocking Signal Reception)
0x0055 TUP BLS (Blocking/Unblocking Signal Sending)
0x0057 TUP CRI (Continuity Recheck Incoming)
0x0058 TUP CRS (Circuit Reset Sending)
0x0059 TUP CGRR (Circuit Group Reset Reception)
0x005A TUP CGRS (Circuit Group Reset Sending)
0x005B TUP MBUS (Maintenance Block/Unblock Sending)
0x005C TUP MBUR (Maintenance Block/Unblock Receiving)
0x005D TUP HBUS (Hardware Block/Unblock Sending)
0x005E TUP HBUR (Hardware Block/Unblock Receiving)
0x005F TUP SBUS (Software Block/Unblock Sending)
0x0060 TUP SBUR (Software Block/Unblock Receiving)

SSUTR2

0x0011 L3P SSUTR2
0x0052 SSUTR2 CPC (Call Processing Control)
0x0053 SSUTR2 SPRC (Signaling Procedure Control)
0x0054 SSUTR2 BLR (Blocking/Unblocking Signal Reception)
0x0055 SSUTR2 BLS (Blocking/Unblocking Signal Sending)
0x0057 SSUTR2 CRI (Continuity Recheck Incoming)
0x0058 SSUTR2 CRS (Circuit Reset Sending)
0x007C SSUTR2 CRO (Continuity Recheck Outgoing)

BT IUP

0x0011 L3P BT IUP (Interconnect User Part)
0x0052 BT IUP CPC (Call Processing Control)
0x0053 BT IUP SPRC (Signaling Procedure Control)

MTP3

0x002B MTP3 HMDT (Message Distribution)
0x002C MTP3 HMRT (Message Routing)
0x002D MTP3 LLSC (Link Set Control)
0x002E MTP3 LSAC (Signaling Link Activity Control)
0x0032 MTP3 RCAT (Signaling Route Set Congestion Test Control)
0x0033 MTP3 RSRT (Signaling Route Set Test Control)
0x0034 MTP3 RTAC (Transfer Allowed Control)
0x0035 MTP3 RTCC (Transfer Controlled Control)
0x0036 MTP3 RTPC (Transfer Prohibited Control)
0x0037 MTP3 RTRC (Transfer Restricted Control)
0x0038 MTP3 TCBC (Changeback Control)
0x0039 MTP3 TCOG (Changeover Control)
0x003A MTP3 TCRC (Controlled Rerouting Control)
0x003C MTP3 TLAC (Link Availability Control)
0x003D MTP3 TPRC (Signaling Point Restart Control)
0x003E MTP3 TRCC (Signaling Route Set Congestion Control)
0x003F MTP3 TSFC (Signaling Traffic Flow Control)
0x0040 MTP3 TSRC (Signaling Route Control)
0x0041 MTP3 SLTC (Signaling Link Test Control)

MTP2

0x0020 MTP2 AERM (Alignment Error Rate Monitor)
0x0021 MTP2 CC (Congestion Control)
0x0022 MTP2 IAC (Initial Alignment Control)
0x0023 MTP2 LSC (Link State Control)
0x0024 MTP2 RC (Reception Control)
0x0025 MTP2 SUERM (Signal Unit Error Rate Monitor)
0x0026 MTP2 TXC (Transmission Control)

SCCP

0x0065 SCLC (SCCP Connectionless Control)

0x0066 SCRC (SCCP Routing Control)

0x0067 SUSI (SCCP User Interface)

0x0068 SPPC (Signaling Point Prohibited Control)

0x0069 SPAC (Signaling Point Allowed Control)

0x006A SPCC (Signaling Point Congestion Control)

0x006B SSPC (Subsystem Prohibited Control)

0x006C SSAC (Subsystem Allowed Control)

0x006D SSTC (Subsystem Status Test Control)

0x6EB CST (Broadcast)

0x006F LBCS (Local Broadcast)

TCAP

0x0070 TUSI (TCAP User Interface)
0x0071 CCO (Component Portion Control)
0x0072 ISM (Component Portion)
0x0073 TCO (Transaction Coordinator)
0x0074 TSM (Transaction State Machine)
0x0075 DHA (Dialog Handling)

ISDN

0x0005 L3P Call Control*
0x0006 L3P D Channel Control
0x0007 L3P B Channel Control
0x0008 L3 Call Reference
0x0009 L3 Global Call Reference
0x000A L3 D Channel Control
0x000B L3 B Channel Control
0x0091 V5 L3P PSTN
0x0092 V5 L3P BCC
0x0093 V5 L3P Manager

DASS2/DPNSS

0x000D DASS2/DPNSS L3P Call Control*
0x0051 DASS2/DPNSS Virtual Call Control*

H.323

0x00A0 L3P RAS
0x00A1 L3P H.225
0x00A2 L3P H.245

SIP

0x00A7 SIP UA

VoIP

0x009C VDAC-ONE VoIP Component*
0x009F - IP Network Interface Series 2 VoIP Component *

PPL Component Addressing

Overview This section lists the PPL Components and the associated AIBs used for PPL messages.

Guidelines The following are guidelines for the information in the tables later in this section.

- PPL timer and configuration bytes are stored on a *per channel* basis.
- The *PPL Transmit Signal Configure* message (0xD2) is applies to the E1 and T1 protocols.

The tables later in this section show the AIBs used to address the various PPL components in these PPL-related API messages listed below:

- *PPL Assign* (0xD1)
- *PPL Audit Configure* (0xDC)
- *PPL Audit Query* (0xDD)
- *PPL Configure* (0xD7)
- *PPL Data Query* (0xDE)
- *PPL Event Indication* (0x43)
- *PPL Event Request* (0x44)
- *PPL Timer Configure* (0xCF)

The *PPL Transmit Signal Configure* message is not included in the tables because it uses the Channel address AIB only.

For span/channel addressed state machines, the following messages can use either a single entity or range address method:

- *PPL Assign* (0xD1)
- *PPL Configure* (0xD7)
- *PPL Timer Configure* (0xCF)
- *PPL Transmit Signal Configure* (0xD2)

For span/channel addressed state machines, the following messages use a single entity address method only:

- *PPL Audit Query* (0xDD)
- *PPL Data Query* (0xDE)

- *PPL Event Indication* (0x43)
- *PPL Event Request* (0x44)

For information on AIB formats, refer to the *Address Elements* chapter.

PPL Events

The *PPL Event Indication* and *PPL Event Request* messages are used to send and receive additional call processing data about the call.

PPL Event Request

This message is initiated by the host, and sends an external event directly to the specified PPL component.

PPL Event Indication

This message enables the PPL component on the CSP to report external events to the host application.

The *PPL Event Request* and *PPL Event Indication* messages must include the PPL component and event. They can include optional data in an ICB.

ICB subtype values can include Layer 3 (ISDN, SS7, SIP, or H.323) parameters. Refer to the *Information Control Blocks* chapter.

E1/T1

Comp. ID	Comp. Name	PPL Assign	PPL Audit Config	PPL Audit Query	PPL Config	PPL Data Query	PPL Event Request	PPL Timer Config
0x0001/ 0x0003	E1/T1	Channel (0x0D)						

**Layer 4/Layer 5 Call Control
PPL Information**

Comp. ID	Comp. Name	AIBs for PPL Messages						
		PPL Assign	PPL Audit Config	PPL Audit Query	PPL Config	PPL Data Query	PPL Event Request	PPL Timer Config
0x0061	CH - Channel Management	Channel (API: 0x0D)						
0x0062	CM - Call Management	Channel (Extended API: 0x0D)						
0x0063	PC - Physical Connection	Channel (Extended API: 0x0D)	N/A	Channel (Extended API: 0x0D)				
0x0064	RTR - Router	N/A		Router (0x29)			N/A	Router (0x29)
0x0086	MCC Group	Router (0x29)		Router (0x29)				
0x0087	MCC Channel	Router (0x29)		Router (0x29)				

SS7

Comp. ID	Comp Name	PPL Assign	PPL Audit Config	PPL Audit Query	PPL Config	PPL Data Query	PPL Event Req	PPL Event Ind	PPL Timer Config
0x000F	L3P CIC	Channel (0x0D)					CIC (0x3C)/ Channel (0x0D)		Channel (0x0D)
0x0010	L3P Link	Link	N/A	Link (0x09)					
0x0012	ISUP CPC	Channel (0x0D)							
0x0013	ISUP SPRC	Stack (0x08)	N/A	Stack (0x08)				Stack (0x08)/ Channel (0x0D)	Stack (0x08)
								Channel (0x0D)	

Comp. ID	Comp Name	PPL Assign	PPL Audit Config	PPL Audit Query	PPL Config	PPL Data Query	PPL Event Req	PPL Event Ind	PPL Timer Config
0x0014	ISUP CCI	Channel (0x0D)	N/A	Channel (0x0D)					
0x0015	ISUP CRI								
0x0016	ISUP BLS								
0x0017	ISUP BLR								
0x0018	ISUP CRS								
0x0019	ISUP CRR								
0x001A	ISUP UCIC								
0x0042	ISUP HLB								
0x0043	ISUP HRB								
0x0044	ISUP CRCR								

Comp ID	Comp Name	PPL Assign	PPL Audit Config	PPL Audit Query	PPL Config	PPL Data Query	PPL Event Req	PPL Event Ind	PPL Timer Config
0x001B	ISUP CGRS	N/A	N/A	Channel (0x0D)			N/A		Channel (0x0D)
0x001C	ISUP CGRR								
0x001D	ISUP GBUS								
0x001E	ISUP GBUR								
0x001F	ISUP HGBR								
0x0045	ISUP MGBS								
0x0046	ISUP MGBR								
0x0047	ISUP HGBS								
0x0076	ISUP CQS								
0x0077	ISUP CQR								
0x0078	ISUP CVS	Channel (0x0D)		Channel (0x0D)					
0x0079	ISUP CVR								
0x0081	ISUP CRCS								
0x0082	ISUP DCO								
0x0083	ISUP CRO								
0x0085	ISUP SSC								

Comp ID	Comp Name	PPL Assign	PPL Audit Config	PPL Audit Query	PPL Config	PPL Data Query	PPL Event Req	PPL Event Ind	PPL Timer Config	
0x00A8	ISUP ACC	N/A	N/A	N/A	SS7 Destination (0x1C)	N/A			SS7 Destination (0x1C)	
0x0011	L3P TUP	Channel (0x0D)		Channel (0x0D)						
0x0052	TUP CPC									
0x0053	TUP SPRC	Stack (0x08)		Stack (0x08)						
0x0054	TUP BLR	Channel (0x0D)		Channel (0x0D)						
0x0055	TUP BLS									
0x0057	TUP CRI									
0x0058	TUP CRS									
0x0059	TUP CGRR	N/A								
0x005A	TUP CGRS									
0x005B	TUP MBUS									
0x005C	TUP MBUR									
0x005D	TUP HBUS									
0x005E	TUP HBUR									
0x005F	TUP SBUS									
0x0060	TUP SBUR									

Comp. ID	Comp Name	PPL Assign	PPL Audit Config	PPL Audit Query	PPL Config	PPL Data Query	PPL Event Req	PPL Event Ind	PPL Timer Config
0x0020	MTP2 AERM	Link (0x09)	N/A						
0x0021	MTP2 CC								
0x0022	MTP2 IAC								
0x0023	MTP2 LSC								
0x0024	MTP2 RC								
0x0025	MTP2 SURM								
0x0026	MTP2 TXC								
0x002B	MTP3 HMDT *	Stack (0x08)							
0x002C	MTP3 HMRT **								
0x002D	MTP3 LLSC								

Comp. ID	Comp Name	PPL Assign	PPL Audit Config	PPL Audit Query	PPL Config	PPL Data Query	PPL Event Req	PPL Event Ind	PPL Timer Config
<p>Notes:</p> <p>* When events 0x0005 Receive MSU Discarded and 0x0006 UPU MSU Received are received, the AIB is Destination (0x1C).</p> <p>** When events 0x0003 Q.752 Destination Report, 0x0005 Transmit MSU Discarded and, 0x0006 UPU MSU Transmitted are received, the AIB is Destination (0x1C).</p>									

Comp ID	Comp Name	PPL Assign	PPL Audit Config	PPL Audit Query	PPL Config	PPL Data Query	PPL Event Req	PPL Event Ind	PPL Timer Config
0x002E	MTP3 LSAC	Link (0x09)	N/A			Link (0x09)			
0x0032	MTP3 RCAT	Destination (0x1C)				Destination (0x1C)			
0x0033	MTP3 RSRT	Route (0x1D)				Route (0x1D)			
0x0034	MTP3 RTAC						Route (0x1D)		
0x0035	MTP3 RTCC	Stack (0x08)				Stack (0x08)			
0x0036	MTP3 RTPC	Route (0x1D)				Route (0x1D)			
0x0037	MTP3 RTRC	Stack (0x08)				Stack (0x08)			
0x0038	MTP3 TCBC	Link (0x09)				Link (0x09)			
0x0039	MTP3 TCOC						Link (0x09)		
0x003A	MTP3 TCRC	Route (0x1D)				Route (0x1D)			
0x003C	MTP3 TLAC	Link (0x09)				Link (0x09)			
0x003D	MTP3 TPRC	Stack (0x08)				Stack (0x08)			
0x003E	MTP3 TRCC	Destination (0x1C)				Destination (0x1C)			
0x003F	MTP3 TSFC *	Stack (0x08)				Stack (0x08)			
0x0040	MTP3 TSRC						Stack (0x08)		

Comp. ID	Comp Name	PPL Assign	PPL Audit Config	PPL Audit Query	PPL Config	PPL Data Query	PPL Event Req	PPL Event Ind	PPL Timer Config
<p>* When events 0x0001 Destination Inaccessible and 0x0002 Destination Accessible are received, the AIB is Destination (0x1C).</p>									

Comp ID	Comp Name	PPL Assign	PPL Audit Config	PPL Audit Query	PPL Config	PPL Data Query	PPL Event Req	PPL Event Ind	PPL Timer Config																																																																																		
0x0041	MTP3 SLTC	Link (0x09)	N/A					Link (0x09)																																																																																			
0x0065	SCCP SCLC	Stack (0x08)						N/A					N/A																																																																														
0x0066	SCCP SCRC												N/A						Subsystem Number (0x2A)																																																																								
0x0067	SCCP SUSI																		N/A						N/A																																																																		
0x0068	SCCP SPPC																								N/A						N/A																																																												
0x0069	SCCP SPAC																														N/A						N/A																																																						
0x006A	SCCP SPCC																																				N/A						N/A																																																
0x006B	SCCP SSPC																																										N/A						N/A																																										
0x006C	SCCP SSAC																																																N/A						N/A																																				
0x006D	SCCP SSTC																																																						N/A						N/A																														
0x006E	SCCP BCST																																																												N/A						N/A																								
0x006F	SCCP LBCS																																																																		N/A						N/A																		
0x0070	TCAP TUSI																																																																								N/A						Subsystem Number (0x2A)												
0x0071	TCAP CCO																																																																														N/A						N/A						
0x0072	TCAP ISM																																																																																				N/A						N/A
0x0073	TCAP TCO	Stack (0x08)																																																																																									
0x0074	TCAP TSM	N/A																																																																																									

ISDN

Comp ID	Comp. Name	PPL Assign	PPL Audit Config.	PPL Audit Query	PPL Config.	PPL Data Query	PPL Event Request	PPL Timer Config.
0x0005	L3P CC	Channel, D (0x0D) †	Channel, B (0x0D)		Channel, D (0x0D)		Channel, B (0x0D)	Channel, D (0x0D)
0x0006	L3P DSM	Channel, D only (0x0D)	N/A	Channel, D only (0x0D)				
0x0007	L3P BSM	Channel, B only (0x0D)		Channel, B only (0x0D)				
0x0008	L3 CR	Channel, D (0x0D) †	Channel, B (0x0D)		Channel, D (0x0D)		Channel, B (0x0D)	Channel, D (0x0D)
0x0009	L3 GCR	Channel, D only (0x0D)	N/A	Channel, D only (0x0D)				
0x000A	L3 DSM	Channel, D only (0x0D)		Channel, D only (0x0D)				
0x000B	L3 BSM	Channel, D (0x0D) †		Channel, B (0x0D)	Channel, D (0x0D)		Channel, B (0x0D)	Channel, D (0x0D))

† In these cases, the PPL components are instantiated on a per call basis and the PPL assignment is made to the controlling D channel. All subsequent calls then use the newly assigned L3P CR and/or L3 CR component.

DASS2/DPNSS

Comp. ID	Comp. Name	PPL Assign	PPL Audit Config	PPL Audit Query	PPL Config	PPL Data Query	PPL Event Request	PPL Timer Config
0x000D	DASS2/ DPNSS L3P CC	Channel, B (0x0D)	N/A		Channel, B (0x0D)			
0x000E	DASS2/ DPNSS L3P D Channel	Channel, D only (0x0D)		Channel, D only (0x0D)				
0x0051	DPNSS L3P VCC	Channel, B (0x0D)		Channel, B (0x0D)				

H.323

Comp. ID	Comp. Name	PPL Assign	PPL Audit Config.	PPL Audit	PPL Configure	PPL Data Query	PPL Event Request/ Indication	PPL Timer Config
0x00A0	L3P RAS	Channel (0x0D)	N/A	Channel (0x0D)				
0x00A1	L3P H.225	Channel (0x0D)		Channel (0x0D)				
0x00A2	L3P H.245	Channel (0x0D)		Channel (0x0D)				

Important! AIB (00 01 0d 03 ff ff ff) is valid for the following PPL Event Requests and Indications which are not related to any active calls: RRQ, URQ, RCF, UCF, RRQ FAIL, URQ FAIL, RRJ, URJ.

Other PPL Event Requests and Indications have a valid Span Channel.

SIP

Comp ID	Comp. Name	PPL Assign	PPL Audit Config.	PPL Audit Query	PPL Config.	PPL Data Query	PPL Event Request/ Indication	PPL Timer Config.
0x00A7	SIP UA	N/A					0x7F *	N/A
							0x0D **	
<p>* Used for registration events</p> <p>** Used for call control events</p> <p>For more information, refer to PPL Information: SIP UA 0xA7 in the <i>SIP Software</i> chapter in the <i>Developer's Guide: Internet Protocol</i>.</p>								

VoIP

Comp. ID	Comp. Name	PPL Assign	PPL Audit Config.	PPL Audit Query	PPL Config.	PPL Data Query	PPL Event Request/ Indication	PPL Timer Config.
0x009C	VDAC-ONE VoIP	N/A					Channel 0x0D	N/A
0x009F	IP Network Interface Series 2 VoIP						Channel 0x0D	N/A

7 Timers and Filters

Overview

Purpose This chapter lists the Timers and Filters used in the EXS API for the CSP.

The CSP expects timers in API messages to be configured in 10-millisecond units, Most Significant Byte (MSB) first and Least Significant Byte (LSB) second.

To determine a timer of 10 seconds, you must convert the time to millisecond units (s = seconds, ms = milliseconds):

$10\text{ s} = 10,000\text{ ms} = 1,000\text{ 10-ms units}$

$1,000 = 3E8$ in hexadecimal, which is expressed as 0x03, 0xE8.

You can determine timers using the conversion chart in the following section.

Timer Conversion Table

Use the table below to convert times into the format explained above.

The length of your timer is listed in the *Time* column and the timer's hexadecimal equivalent is listed in the *MSB* and *LSB* columns.

Time	MSB	LSB
50 ms	0x00	0x05
100 ms	0x00	0x0A
150 ms	0x00	0x0F
200 ms	0x00	0x14
250 ms	0x00	0x19
300 ms	0x00	0x1E
350 ms	0x00	0x23
400 ms	0x00	0x28
450 ms	0x00	0x2D
500 ms	0x00	0x32
550 ms	0x00	0x37
600 ms	0x00	0x3C
650 ms	0x00	0x41
700 ms	0x00	0x46
750 ms	0x00	0x4B
800 ms	0x00	0x50
850 ms	0x00	0x55
900 ms	0x00	0x5A
950 ms	0x00	0x5F
1050 ms	0x00	0x69
1100 ms	0x00	0x6E
1150 ms	0x00	0x73
1200 ms	0x00	0x78

Time	MSB	LSB
1250 ms	0x00	0x7D
1300 ms	0x00	0x82
1350 ms	0x00	0x87
1400 ms	0x00	0x8C
1450 ms	0x00	0x91
1500 ms	0x00	0x96
1550 ms	0x00	0x9B
1600 ms	0x00	0xA0
1650 ms	0x00	0xA5
1700 ms	0x00	0xAA
1750 ms	0x00	0xAF
1800 ms	0x00	0xB4
1850 ms	0x00	0xB9
1900 ms	0x00	0xBE
1950 ms	0x00	0xC3
2000 ms	0x00	0xC8
1 s	0x00	0x64
2 s	0x00	0xC8
3 s	0x01	0x2C
4 s	0x01	0x90
5 s	0x01	0xF4
6 s	0x02	0x58
7 s	0x02	0xBC
8 s	0x03	0x20
9 s	0x03	0x84
10 s	0x03	0xE8
11 s	0x04	0x4C
12 s	0x04	0xB0
13 s	0x05	0x14

Time	MSB	LSB
14 s	0x05	0x78
15 s	0x05	0xDC
16 s	0x06	0x40
17 s	0x06	0xA4
18 s	0x07	0x08
19 s	0x07	0x6C
20 s	0x07	0xD0
21 s	0x08	0x34
22 s	0x08	0x98
23 s	0x08	0xFC
24 s	0x09	0x60
25 s	0x09	0xC4
26 s	0x0A	0x28
27 s	0x0A	0x8C
28 s	0x0A	0xF0
29 s	0x0B	0x54
30 s	0x0B	0xB8
31 s	0x0C	0x1C
32 s	0x0C	0x80
33 s	0x0C	0xE4
34 s	0x0D	0x48
35 s	0x0D	0xAC
36 s	0x0E	0x10
37 s	0x0E	0x74
38 s	0x0E	0xD8
39 s	0x0F	0x3C

T1 Filters

The *Filter/Timer Configure* message modifies approximately 100 combined timers and filters. There are approximately 14 signal scanning filters, 63 signal scanning timers, and 36 transmit signal timers for T1 trunk types.

Filter values are in 10 millisecond decimal units. If there is no value in a field, the filter does not apply.

Signaling Scanning Filters

Signal Scanning Filters (table below) define the duration that an expected signaling condition is expected to be valid before it is declared to have occurred.

ID	PPL Timer No.	Signal Scanning Filter	E&M	FXO LS	FXS LS	FXO GS	FXS GS
0x01	0x01	Preseize	10			10	
0x02	0x02	Inseize	15	600	20	600	100
0x03	0x03	Modified Incoming Release	70	500	70	60	70
0x04	0x04	Modified Outgoing Release	70		70	60	70
0x05	0x05	Normal Incoming Release	70		70	60	70
0x06	0x06	Normal Incoming Release w/Flash	102		102	102	102
0x07	0x07	Normal Outgoing Release	70		70	60	70
0x08	0x08	Normal Outgoing Release w/Flash	102		102	102	102
0x09	0x09	Post Inseize Acknowledgment					
0x0A	0x0A	Outseize Acknowledgment	10		18	10	18
0x0B	0x0B	Outseize Answer	35				
0x0C	0x0C	Outseize Dial Signal End	4				
0x0D	0x0D	First Release	70		70	15	
0x0E	0x0E	ANI Req Offhook	20		20		20

Signaling Scanning Timers Signal Scanning Timers (table below) define the minimum and maximum acceptable durations for signaling condition to exist from the point they are declared valid.

Timer values are in 10 millisecond decimal units. If there is no value in a field, the timer does not apply.

ID	PPL Timer No.	Signal Scanning Timer	E&M	FXO LS	FXS LS	FXO GS	FXS GS
0x01	0x15	Inseize Complete				604	
0x02	0x16	Post Inseize Complete					22
0x03	0x17	Start Normal Outgoing Release	0				
0x04	0x18	Start Normal Incoming Release	0				
0x05	0x19	Outseizure Acknowledgment	500			500	600
0x06	0x1A	Outseizure Answer	10,000		64,000		
0x07	0x1B	Guard Time-out	70	85		85	
0x08	0x1C	Release	3,200		3,200	3,200	3,200
0x09	0x1D	Glare Detection	10	30		10	
0x0A	0x1E	Minimum Flash	30	30	30	30	30
0x0B	0x1F	Maximum Flash	100	100	100	100	100
0x0C	0x20	Start Dial (Wink 1) Minimum Wink	10				
0x0D	0x21	Start Dial (Wink 2) Maximum Wink	35				
0x0E	0x22	Minimum Delay Dial Signal	14				
0x0F	0x23	Maximum Delay Dial Signal	400				
0x10	0x24	Post Start Dial Signal Outpulse Delay	8				
0x11	0x25	Wink 2 Minimum Receive Duration	10				
0x12	0x26	Wink 2 Maximum Receive Duration	50				
0x13	0x27	Wink 2 Minimum Post Outpulse Delay	10				

ID	PPL Timer No.	Signal Scanning Timer	E&M	FXO LS	FXS LS	FXO GS	FXS GS
0x14	0x28	Wink 2 Maximum Post Outputpulse Delay	350				
0x15	0x29	Wink 2 Maximum Receive Detection	1,150				
0x16	0x2A	Wink 3 Minimum Receive Duration	10				
0x17	0x2B	Wink 3 Maximum Receive Duration	60				
0x18	0x2C	Wink 3 Minimum Post Outputpulse Delay	10				
0x19	0x2D	Wink 3 Maximum Post Outputpulse Delay	350				
0x1A	0x2E	Wink 3 Maximum Receive Detection	400				
0x1B	0x2F	Wink 4 Minimum Receive Duration	10				
0x1C	0x30	Wink 4 Maximum Receive Duration	35				
0x1D	0x31	Wink 4 Minimum Post Outputpulse Delay	10				
0x1E	0x32	Wink 4 Maximum Post Outputpulse Delay	350				
0x1F	0x33	Wink 4 Maximum Receive Detection	400				
0x20	0x34	Wink 5 Minimum Receive Duration	10				
0x21	0x35	Wink 5 Maximum Receive Duration	35				
0x22	0x36	Wink 5 Minimum Post Outputpulse Delay	10				
0x23	0x37	Wink 5 Maximum Post Outputpulse Delay	350				
0x24	0x38	Wink 5 Maximum Receive Detection	400				

ID	PPL Timer No.	Signal Scanning Timer	E&M	FXO LS	FXS LS	FXO GS	FXS GS
0x25	0x39	Wink 6 Minimum Receive Duration	10				
0x26	0x3A	Wink 6 Maximum Receive Duration	35				
0x27	0x3B	Wink 6 Minimum Post Outputpulse Delay	10				
0x28	0x3C	Wink 6 Maximum Post Outputpulse Delay	350				
0x29	0x3D	Wink 6 Maximum Receive Detection	400				
0x2A	0x3E	Wink 7 Minimum Receive Duration	10				
0x2B	0x3F	Wink 7 Maximum Receive Duration	35				
0x2C	0x40	Wink 7 Minimum Post Outputpulse Delay	10				
0x2D	0x41	Wink 7 Maximum Post Outputpulse Delay	350				
0x2E	0x42	Wink 7 Maximum Receive Detection	400				
0x2F	0x43	Wink 8 Minimum Receive Duration	10				
0x30	0x44	Wink 8 Maximum Receive Duration	35				
0x31	0x45	Wink 8 Minimum Post Outputpulse Delay	10				
0x32	0x46	Wink 8 Maximum Post Outputpulse Delay	350				
0x33	0x47	Wink 8 Maximum Receive Detection	400				
0x34	0x48	Post ANI Offhook Outputpulse Detection	100				
0x35	0x49	Maximum Receive ANI Off-hook Request	1,600				

ID	PPL Timer No.	Signal Scanning Timer	E&M	FXO LS	FXS LS	FXO GS	FXS GS
0x36	0x4A	Maximum Receive Dialtone Detection	800	800	800	800	800
0x37	0x4B	MFR1 Minimum Receive KP Duration	5	5	5	5	5
0x38	0x4C	MFR1 Minimum Receive Digit Duration	3	3	3	3	3
0x39	0x4D	MFR1 Maximum Receive KP Duration	1,000	1,000	1,000	1,000	1,000
0x3A	0x4E	MFR1 Minimum Receive Interdigit Duration	3	3	3	3	3
0x3B	0x4F	MFR1 Maximum Receive Interdigit Duration	1,000	1,000	1,000	1,000	1,000
0x3C *	0x50	DTMF Maximum Receive 1st Digit Detection	2,000	2,000	2,000	2,000	2,000
0x3D	0x51	DTMF Minimum Receive Digit Duration	4	4	4	4	4
0x3E	0x52	DTMF Minimum Receive Interdigit Duration	4	4	4	4	4
0x3F	0x53	DTMF Maximum Receive Interdigit Duration	2,000	2,000	2,000	2,000	2,000

Important! * To configure the DTMF timers for H.323 use *PPL Timer Configure 0x00CF* and then refer to the *API Developers Guide: IP, H.323 Chapter*.

Transmit Signaling Timers

Transmit Signaling Timers (table below) define the durations used when the system is generating signaling or tones. With the T-ONE card, timer values can be modified using either the PPL Timer Configure message

Timer values are in 10 millisecond decimal units. If there is no value in a field, the timer does not apply.

ID	PPL Timer No.	Transmit Signaling Timer	E&M	FXO LS	FXS LS	FXO GS	FXS GS
0x01	0x5B	Wink (1 or KP) Duration	20				
0x02	0x5C	Flash Duration	60	60	60	60	60
0x03	0x5D	Primary Outseize Signaling					
0x04	0x5E	Outseize Signaling Complete	30	30			
0x05	0x5F	Delay Dial Signal Start	7				
0x06	0x60	Fixed Pause	7	7		7	7
0x07	0x61	Timed Answer		50		50	
0x08	0x62	Ringing On Duration			200		
0x09	0x63	Ringing Off Duration			400		
0x0A	0x64	Wink 1 Minimum Transmit Delay	8				
0x0B	0x65	Wink 2 Minimum Transmit Delay	80				
0x0C	0x66	Wink 2 Maximum Transmit Delay	1,150				
0x0D	0x67	Wink 2 Transmit Duration	20				
0x0E	0x68	Wink 3 Minimum Transmit Delay	10				
0x0F	0x69	Wink 3 Maximum Transmit Delay	40				
0x10	0x6A	Wink 3 Transmit Duration	20				
0x11	0x6B	Wink 4 Minimum Transmit Delay	8				

ID	PPL Timer No.	Transmit Signaling Timer	E&M	FXO LS	FXS LS	FXO GS	FXS GS
0x12	0x6C	Wink 4 Maximum Transmit Delay	40				
0x13	0x6D	Wink 4 Transmit Duration	20				
0x14	0x6E	Wink 5 Minimum Transmit Delay	8				
0x15	0x6F	Wink 5 Maximum Transmit Delay	40				
0x16	0x70	Wink 5 Transmit Duration	20				
0x17	0x71	Wink 6 Minimum Transmit Delay	8				
0x18	0x72	Wink 6 Maximum Transmit Delay	40				
0x19	0x73	Wink 6 Transmit Duration	20				
0x1A	0x74	Wink 7 Minimum Transmit Delay	8				
0x1B	0x75	Wink 7 Maximum Transmit Delay	40				
0x1C	0x76	Wink 7 Transmit Duration	20				
0x1D	0x77	Wink 8 Minimum Transmit Delay	8				
0x1E	0x78	Wink 8 Maximum Transmit Delay	40				
0x1F	0x79	Wink 8 Transmit Duration	20				
0x20	0x7A	MFR1 Transmit KP Duration	10	10	10	10	10
0x21	0x7B	MFR1 Transmit Digit Duration	6	6	6	6	6
0x22	0x7C	MFR1 Transmit Interdigit Duration	6	6	6	6	6
0x23	0x7D	DTMF Transmit Digit Duration	6	6	6	6	6
0x24	0x7E	DTMF Transmit Interdigit Duration	6	6	6	6	6

E1 Timers

The table below shows the timers used with the E1 component (0x01).

Timer ID	Description: (State Used); Timer Description	Default Value (10 ms)
1	(S19) Incoming Layer 4 busy out Acknowledge wait	400
2	(S2) Incoming Stage 1 FWD digit receive wait	2000
3	(S3) Incoming Stage 2 FWD digit receive wait	2000
4	(S4) Incoming Stage 3 FWD digit receive wait	2000
5	(S5) Incoming Stage 4 FWD digit receive wait	2000
6	(S13) Layer 4 Connect wait	24,000
7	(S14) Layer 4 wait	400
8	(S21) Clear Forward (idle line signaling) wait	1500
9	(S3) Incoming Stage 2 Initial FWD digit receive wait	2000
10	(S2) Incoming Stage 1 Initial FWD digit receive wait	2000
11	(S5) Incoming Stage 4 Initial FWD digit receive wait	2000
12	(S4) Incoming Stage 3 Initial FWD digit receive wait	2000
13	(S15) Layer 3 Circuit Release wait	10
14	(S18) Layer 3 Circuit Release wait	10
15	(S6) Host Control Wait	6000
16	(S13) Layer 3 Connect wait	24000
17	(S29) Seize Acknowledge wait	400
18	(S31) Time to wait before transmitting idle signal after detecting glare	10
19	(S32) Time to wait before looking for idle signal after transmitting idle signal after detecting glare	10
20	(S39) Outgoing Stage 1 BWD R2 signal wait	1500
21	(S39) Outgoing Stage 1 Initial BWD R2 signal wait	1500
22	(S41) Outgoing Stage 2 BWD R2 signal wait	1500
23	(S41) Outgoing Stage 2 Initial BWD R2 signal wait	1500

Timer ID	Description: (State Used); Timer Description	Default Value (10 ms)
24	(S43) Outgoing Stage 3 BWD R2 signal wait	1500
25	(S43) Outgoing Stage 3 Initial BWD R2 signal wait	1500
26	(S45) Outgoing Stage 4 BWD R2 signal wait	1500
27	(S45) Outgoing Stage 4 Initial BWD R2 signal wait	1500
28	(S46) Outgoing wait for answer	24000
29	(S48) Outgoing Layer 4 Clear wait	400
30	(S49) Outgoing idle line signal wait	12000
31	(S50) Outgoing idle line signal wait after glare detected	12000
32	(S24) Host Control wait	24000
33	(S33) Time to wait for Clear Forward after glare detected	1000
34	Receive line signaling filter	4
36	(S6, S24) Incoming BWD R2 Cycle Complete Event wait	400
38	(S10) Time to wait for idle line signaling after invalid line signal detected while receiving FWD R2 digits	2000
39	(S22) Layer 3 Clear Forward wait	300
40	(S39) Outgoing Stage 1 BWD A signal wait after sending all of Stage 1 FWD digits	2000
41	(S2, S3, S4, S5) FWD R2 digits receive wait when String Collection Data field in Inpulsing Parameters Configure is set to 0xFF	144
43	R2 Cycle Complete Event wait	200
45	(S43) Time to wait for next BWD R2 signal when stage 3 digits have been outpulsed	2000