Dialogic
Making Innovation Thrive

Dialogic® Diva® API for .NET

Developer's Reference Guide

Part of the Dialogic® Diva® Software Development Kit

August 2009 206-472-04

www.dialogic.com

Copyright and Legal Disclaimer

All contents of this document are furnished for informational use only and are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation or its subsidiaries ("Dialogic"). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Dialogic does not warrant the accuracy of this information and cannot accept responsibility for errors, inaccuracies 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 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 may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic does 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 and no such licenses are provided except pursuant to a signed agreement with Dialogic. More detailed information about such intellectual property is available from Dialogic's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. **Dialogic 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, Realblocs, Realcomm 100, NetAccess, Instant ISDN, TRXStream, Exnet, Exnet Connect, EXS, ExchangePlus VSE, Switchkit, N20, Powering The Service-Ready Network, Vantage, Making Innovation Thrive, Connecting People to Information, Connecting to Growth and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

Microsoft, Windows, and Visual Studio are 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.

To contact Dialogic Customer Support, visit our Web site at www.dialogic.com/support.

This software is based in part on the work of the Independent JPG Group.

Tiff Lib
Copyright © 1988-1997 Sam Leffler
Copyright © 1991-1997 Silicon Graphics, Inc.

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that (i) the above copyright notices and this permission notice appear in all copies of the software and related documentation, and (ii) the names of Sam Leffler and Silicon Graphics may not be used in any advertising or publicity relating to the software without the specific, prior written permission of Sam Leffler and Silicon Graphics.

Dialogic Corporation License Agreement for Use of Software

This is an Agreement between you, the Company, and your Affiliates (referred to in some instances as "You" and in other instances as "Company") and all your Authorized Users and Dialogic Corporation ("Dialogic").

YOU SHOULD CAREFULLY READ THE SOFTWARE LICENSE AGREEMENT ("AGREEMENT") ON THIS SEALED PACKAGE BEFORE OPENING THE PACKAGE. BY OPENING THE PACKAGE, YOU ACCEPT THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE WITH OR ARE UNWILLING TO ACCEPT THESE TERMS AND CONDITIONS, YOU MAY RETURN THE PACKAGE IN UNOPENED "AS NEW" CONDITION (INCLUDING ALL DOCUMENTATION AND BINDERS OR OTHER CONTAINERS) FOR A FULL REFUND. BY DOWNLOADING, INSTALLING, COPYING OR OTHERWISE USING THE ENCLOSED SOFTWARE ("PROGRAM"), YOU FURTHER AGREE AND ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT AND UNDERSTAND IT, AND THAT BY TAKING ANY ONE OR MORE OF SUCH STEPS/ACTIONS YOU AGREE TO BE BOUND BY SUCH TERMS AND CONDITIONS. DIALOGIC IS UNWILLING TO LICENSE THE SOFTWARE TO YOU IF YOU DO NOT ACCEPT AND AGREE TO BE BOUND BY THE TERMS AND CONDITIONS OF THIS AGREEMENT.

Intellectual Property

The enclosed Software ("Program") and all accompanying documentation are individually and collectively owned by Dialogic Corporation ("Dialogic"), its subsidiaries and/or its suppliers and are protected by all applicable intellectual property laws and international treaty provisions. Therefore, You and Your Authorized Users must treat the Program and documentation like any other material so protected, except as expressly permitted in this Agreement. In particular, but without limitation, You acknowledge that the Program and its accompanying documentation constitute valuable intellectual property rights, including without limitation trade secrets and copyrights, and confidential information of Dialogic. The Program and all programs developed thereunder and all copies thereof (including without limitation translations, compilations, partial copies with modifications and updated works) are proprietary to Dialogic and title to all applicable copyrights, trade secrets, patents and other intellectual property rights therein remains in Dialogic, its subsidiaries, and/or its suppliers. Except as expressly permitted in this Agreement, You shall not sell, transfer, publish, disclose, display or otherwise make available the Program or copies thereof to others. You agree to secure and protect the Program, its accompanying documentation and copies thereof in a manner consistent with the maintenance of Dialogic's rights therein and to take appropriate action by instruction or agreement with Your employees and/or consultants who are permitted access to the Program to satisfy Your obligations hereunder. Violation of any provision of this paragraph shall be the basis for immediate termination of this Agreement. Because unauthorized use or transfer of the Software or documentation may diminish substantially the value of such materials and irrevocably harm Dialogic, if You breach the provisions of this Section of this Agreement, Dialogic shall be entitled to injunctive and/or other equitable relief, in addition to other remedies afforded by law, to prevent a breach of this Section of this Agreement.

Grant of License

Subject to the terms and conditions of this Agreement Dialogic grants to You a non-exclusive, personal, non-transferable license to use the Program in object code form only and solely in accordance with the following terms and conditions:

- You may make, install and use only one (1) copy of the Program on a single-user computer, file server, or on a workstation of a local area network, and only in conjunction with a legally acquired Dialogic® hardware or software product You may also make one copy solely for backup or archive purposes;
- The primary Authorized User on the computer on which the Program is installed may make a second copy for his/her exclusive use on either a home or portable computer;
- You may copy the Program into any machine readable or printed form for backup or modification purposes in support of Your use of one copy of the Program;
- You may distribute the Program in object code only and only as part of, or integrated by You into, a computer system that (i) contains a Dialogic hardware product, (ii) includes a substantial amount of other software and/or hardware manufactured or marketed by You and (iii) is marketed and sublicensed to an end user for the end user's own internal use in the regular course of business (a "Licensed System");
- Each end user to whom a Licensed System is distributed must agree to license terms with respect to the Program that are at least as protective of Dialogic's rights in the Program as those set forth in this Agreement;
- You shall receive one (1) Program master disk, and shall be solely responsible for copying the Program into the Licensed Systems and for warranting the physical media on which it is copied
- You may make one (1) copy of the documentation accompanying the Program, provided that all copyright notices contained within the documentation are retained;
- You may modify the Program and/or merge it into another Program for Your use in one computer; (any portion of this Program will continue to be subject to the terms and conditions of this Agreement);
- You may transfer the Program, documentation and the license to another eligible party within Your Company if the other party agrees to accept the terms and conditions of this Agreement. If You transfer the Program and documentation, You must at the same time either transfer all copies whether in printed or machine readable form to the same party or destroy any copies not transferred; this includes all modifications and portions of the Program contained in or merged into other Programs;

- You shall not remove, and each copy of the Program shall contain, the same copyright, proprietary, patent and/or other applicable intellectual property or other ownership notices, plus any restricted rights legends that appear in the Program and/or this Agreement and, if You copy the Program onto media to which a label may be attached, You shall attach a label to the media that includes all such notices and legends that appear on the Program master disk and envelope;
- You may not rent or lease the Program. You may not reverse engineer, decompile or disassemble the Program. Except as is strictly necessary for You to integrate the Program with other software and/or hardware to produce the Licensed Systems, You shall not copy, modify or reproduce the Program or documentation in any way. You shall use Your best efforts to ensure that any user of the Program does not reverse engineer, decompile or disassemble the Program to derive a source code equivalent of the Program;
- If You transfer possession of any copy, modification or merged portion of the Program or documentation to another party in any way other than as expressly permitted in this Agreement, this license is immediately and automatically terminated;
- The Program may be used only in conjunction with Dialogic hardware;
- The Program shall not be exported or re-exported in violation of any export provisions of the United States or any other applicable jurisdiction.

Upgrades

If the Program is provided as an upgrade and the upgrade is an upgrade from another product licensed to You and Your Authorized Users by Dialogic, the upgrade is governed by the license agreement earlier provided with that software product package and the present Agreement does not grant You additional license(s). If You and Your Authorized Users choose to upgrade this Program or the product used together with the Program and such upgrade requires the license of additional software (whether a charge is associated with such software or not), the license agreement associated with such additional software shall govern the license of such additional software to the exclusion of this Agreement.

Term

The Agreement is effective until terminated. You may terminate it at any time by notifying Dialogic and/or by destroying the Program and all accompanying documentation together with all copies, modifications and merged portions in any form. The Agreement will also terminate automatically upon the occurrence or lack of occurrence of certain terms and/or conditions set forth in this Agreement, or if You fail to comply with any term or condition of this Agreement. You agree that upon any such termination You shall destroy or return to Dialogic the Program and all accompanying documentation supplied by Dialogic, together with any and all copies, modifications and merged portions in any form. All provisions of this Agreement relating to disclaimers of warranties, limitation of liability, remedies, or damages, and licensor's proprietary rights shall survive termination.

Limited Warranty

Dialogic solely warrants the media on which the Program is furnished to You to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of purchase by You as evidenced by a copy of Your receipt. If such a defect appears within the warranty period, You may return the defective media to Dialogic for replacement without charge provided Dialogic, in good faith, determines that it was defective in materials or workmanship. Replacement is Your sole remedy with respect to such a defect. Dialogic offers no warranty for Your reproduction of the Program. This Limited Warranty is void if failure of the Program has resulted from accident, misuse, abuse or misapplication.

Disclaimers, Limitations of Liability and Customer Remedies

Except as set forth in the "Limited Warranty" Section of this Agreement, the Program and accompanying documentation are provided to You "as is." Neither Dialogic, its subsidiaries, its suppliers, nor its licensor(s) (if any) warrants that the Program will meet Your requirements or that its use will be uninterrupted or error-free. Except as set forth in the "Limited Warranty" Section, EACH OF DIALOGIC, ITS SUBSIDIARIES, ITS SUPPLIERS AND ITS LICENSOR(S) (IF ANY) DISCLAIMS ANY AND ALL REPRESENTATIONS AND WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE PROGRAM AND ACCOMPANYING DOCUMENTATION, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR AGAINST LATENT DEFECTS. Except as set forth in the "Limited Warranty" Section, neither Dialogic, its subsidiaries, its suppliers, nor its licensor(s) (if any) shall have any liability to You or any third party for any claim, loss or damage of any kind, including but not limited to lost business profits, business interruption, loss of information, or other pecuniary loss and indirect, punitive, incidental, economic, consequential or special damages, arising out of or in connection with this Agreement and/or the use, inability to use the Program and/or the Program's performance or inability to perform nor from or in connection with the Program's accompanying documentation, or any data or equipment related thereto or used in connection therewith. In no event shall Dialogic's, its subsidiaries', its suppliers' or its licensor(s)'s liability for damages, whether arising out of contract, negligence, warranty, or patent or copyright infringement, exceed the fees You paid for the Program. No representation or warranty regarding the Program may be made without Dialogic's, its subsidiaries', its suppliers', or its licensor(s)'s (if any) prior written consent, and any warranty or representation made by You or Your customers regarding the Program shall not constitute an obligation of Dialogic, its subsidiaries, its suppliers, or other licensor(s) (if any). This limited warranty gives You specific legal rights. You may have other rights, which may vary from jurisdiction to jurisdiction. Also, as some jurisdictions do not allow the exclusion or limitation for certain damages, some of the above limitations may not apply to You.

Right to Audit

If this Program is licensed for use in a Company, Your Company and You individually and collectively agree to keep all usual and proper records and books of accounts and all usual proper entries relating to each installation of the Program during the term of this Agreement and for a period of three (3) years thereafter. During this period, Dialogic may cause an audit to be made of the applicable records in order to verify Your compliance with this Agreement and prompt adjustment shall be made to compensate for any errors or omissions disclosed by such audit. Any such audit shall be conducted by an independent certified public accountant selected by Dialogic and shall be conducted during the regular business hours at Your offices and in such a manner as not to interfere with Your normal business activities. Any such audit shall be paid for by Dialogic unless material discrepancies are disclosed. For such purposes, "material discrepancies" shall mean three percent (3%) or more of the Authorized Users within the Company. If material discrepancies are disclosed,

Your Company agrees to pay Dialogic for the costs associated with the audit as well as the license fees for the additional licensed channels or additional authorized users. In no event shall audits be made more frequently than semi-annually unless the immediately preceding audit disclosed a material discrepancy.

Supplementary Software

Any Supplementary Software provided with the Program and/or referred to in this Agreement is provided "as is" with no warranty of any kind.

Miscellaneous

You acknowledge that You have read this Agreement, that You understand it, and that You agree to be bound by its terms and conditions, and You further agree that this is the complete and exclusive statement of the Agreement between the Dialogic and You ("the Parties"), which supersedes and merges all prior proposals, understandings and all other agreements, oral and written, between the Parties relating to the Program. You agree to indemnify and hold harmless Dialogic and its subsidiaries, affiliates, suppliers, officers, directors and employees from and against any claim, injury, loss or expense, including reasonable attorneys' fees, arising out of (i) Your failure to comply with the provisions of this Agreement, or (ii) any other wrongful conduct by or on behalf of You. This Agreement applies to all updates, future releases, modifications and portions of the Program contained in or merged into other programs. This Agreement may not be modified or altered except by written instrument duly executed by Dialogic. No action, regardless of form, arising out of this Agreement or the use of the Program may be brought by You more than two (2) years after the cause of action has first arisen. Except as provided herein, neither this Agreement nor any rights granted are assignable or transferable, and any assignment or transfer will be null and void. If You authorize any other person to copy the Program, You shall obligate that person in writing to comply with all conditions of this Agreement. Dialogic shall have the right to collect from You its reasonable expenses incurred in enforcing this agreement, including attorney's fees. The waiver or failure of Dialogic to exercise in any respect any right provided for herein shall not be deemed a waiver of any further right hereunder. All rights and remedies, whether conferred hereunder or by any other instrument or law, will be cumulative and may be exercised singularly or concurrently. Failure by either Dialogic or You to enforce any term or condition of the Agreement will not be deemed a waiver of future enforcement of that or any other term or conditions. The terms and conditions stated herein are declared to be severable. Should any term(s) or condition(s) of this Agreement be held to be invalid or unenforceable the validity, construction and enforceability of the remaining terms and conditions of this Agreement shall not be affected. It is expressly agreed that Dialogic and You are acting as independent contractors under this Agreement. These terms and conditions will prevail notwithstanding any different, conflicting or additional terms and conditions that may appear on any other agreement between Dialogic and You. Deviations from these terms and conditions are not valid unless agreed to in writing in advance by an authorized representative of Dialogic. Any notices sent to Dialogic under this Agreement must be sent by registered mail or courier to the attention of Dialogic's legal department at the address below or such other address as may be listed on www.dialogic.com from time to time as being Dialogic's Montreal headquarters.

U.S. Government Restricted Rights

The Program and all accompanying documentation are provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(iii) of The Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraph (c) (1) and (2) of the Commercial Computer Software-Restricted Rights at 48 CFR52.227-19, both as applicable.

Governing Law

Any and all claims arising under this Agreement shall be construed and controlled by the laws in force in the Province of Quebec, Canada, excluding its principles of conflict of laws and the United Nations Convention on Contracts for the Sale of Goods. Dialogic is not obligated under any other agreements unless they are in writing and signed by an authorized representative of Dialogic.

Contractor/ manufacturer is:

Dialogic CORPORATION.

9800 Cavendish Blvd., Montreal, Quebec, Canada H4M 2V9

This Agreement has been drafted in English at the express wish of the parties. Ce contrat a été rédigé en anglais à la demande expresse des parties.

Contents

About This Publication

How to use this online guide

- To view a section, click the corresponding bookmark located on the left.
- To view a topic that contains further information, click the corresponding blue underlined phrase.
- You may wish to print out the pages required for developing your communication application.

Structure of this guide

This guide presents implementation details and functional descriptions of the commands in the Dialogic® Diva® API Library interface. Examples are provided where needed. Constants, data structures, and return codes are also provided.

This guide is structured as follows:

Section	Contents
Dialogic® Diva® SDK Overview	Introduction to the Dialogic® Diva® software development kit and its five application programming interfaces: the Dialogic® Diva® Component API, the Dialogic® Diva®API, the extended CAPI 2.0, the Dialogic® Diva® Management API, and the Dialogic® Diva® API for .NET
Dialogic® Diva® API for .NET Overview	Introduction to the components provided with the Diva API for .NET and prerequisites for using it
Using the .NET Wrapper	Description of naming difference between the Diva API and the Diva API for .NET
Dialogic® Diva® API for .NET Functions	Description of the functions provided with the Diva API for .NET

CHAPTER 1

Dialogic® Diva® SDK Overview

The Dialogic Diva SDK can be used in combination with Dialogic® Diva® Media Boards and Dialogic® Host Media Processing (HMP) software. On these communication platforms, the Diva SDK provides the following application programming interfaces (APIs): the Dialogic® Diva® API, the Dialogic® Diva® Components API, and the Dialogic® Diva® API for .NET. For the Diva Media Boards, two additional APIs are available: the Dialogic® Diva® Management API and the Extended CAPI 2.0.

It is planned that new versions of the Diva SDK will be released periodically, and it is intended that such new versions will be backwards compatible so as to allow applications developed on the basis of earlier versions of the Diva SDK to be used with the new versions.

The Diva SDK includes the following components:

- Libraries providing functions to access the Dialogic® Diva® communication platforms
- DLLs containing the interfaces and component services
- Programming samples in source code
- Documentation explaining the functions of the Diva SDK

The components can be found as follows:

Component	Path
Libraries of the Diva SDK	\SDK\BASIC\LIB\
DLLs of the Diva SDK and compiled samples applications	\SDK\BASIC\BIN
Samples for the Diva SDK	\SDK\BASIC\SAMPLES\
Libraries of the Diva Management API	\SDK\MANAGEMENT\LIB
Samples for the Management API	\SDK\MANAGEMENT\SAMPLES
Documentation	\SDK\DOC\

The Diva SDK is available for download it from the Dialogic web site under http://www.dialogic.com/products/tdm_boards/development_tools/default.htm. If you download the software from the Dialogic web site, extract the files to your hard disk and do not change the directory structure of the extracted files.

The Diva SDK is freely distributed with Dialogic® communication platforms. You do not have to purchase licences for developing applications based on the software development kit.

Dialogic® Diva® SDK application programming interfaces

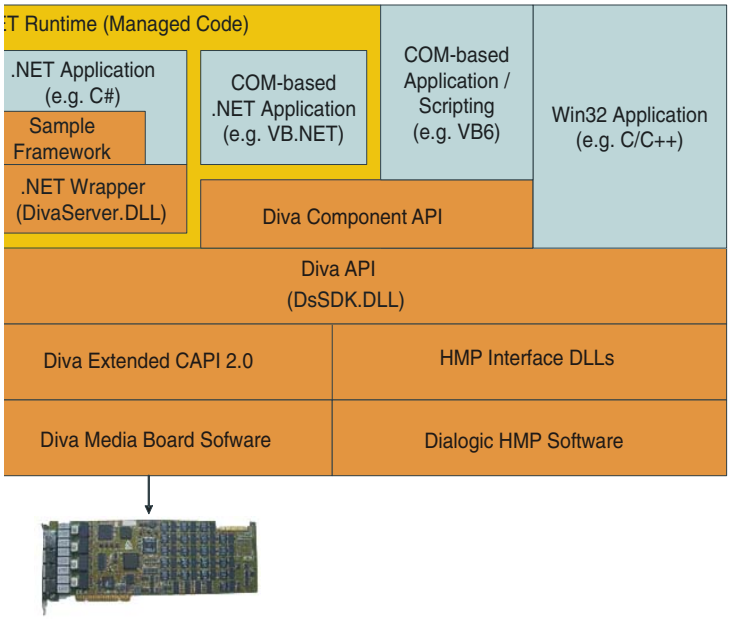
The five application programming interfaces (APIs) of the Dialogic® Diva® SDK represent different layers for the management and development of applications for Dialogic® communication platforms.

- Dialogic® Diva® API: It provides a high-level interface into the communication platforms that allows developers to implement communication applications. It also provides an additional library for data conversion like TIFF to SFF for fax applications.
- Dialogic® Diva® Component API: It provides a set of ActiveX components that allows developers to create new applications or to add telephony and communication features to existing applications. The Component API can be used from scripts and VB.NET and eliminates the need to write directly to a C / C++ API.
- Dialogic® Diva® API for .NET: It is based on the Dialogic® Diva® API and provides access to the Diva API from .NET-based applications.
- Extended CAPI 2.0 (only for Dialogic® Diva® Media Boards): It provides Dialogic-specific CAPI extensions that are fully CAPI 2.0 compliant.

- Dialogic® Diva® Management API: It provides direct hardware access for monitoring, security, and statistics. This API should only be used by applications using CAPI 2.0.

Dialogic® communication platforms provide call control, media streaming, and management functionality that are available on the Diva API, the Diva Component API, and the Diva API for .NET.

The following figure shows the architecture of the programming interfaces and the applications that may access them:



Dialogic® Diva® API

The Diva API is a high-level interface into the Dialogic® communication platforms via a library of "C" function calls. This interface can allow developers to implement various communication applications faster and easier than in the traditional CAPI 2.0 application development.

The Diva API contains modules that can be used as basis for communication applications, such as fax and voice transfer or call control, and in the development of applications for these areas. The modules are intended to be updated so as to offer development bases for additional communication applications.

Even if the Diva API abstracts functions and provides a high level interface, access to low level functions is optionally available. Applications that require access to low level operations, e.g., control over signaling messages, can be performed on the Diva API. This allows existing applications to be extended using the same API, even if the requirements change. The CAPI 2.0 extensions are also available on the Diva API.

The Diva API also allows for access to the management interface of the Dialogic® Diva® Media Board for status and statistic information.

Dialogic® Diva® Component API

The Diva® SDK includes the Diva Component API. This interface provides a set of ActiveX components that allow for creating new applications or for adding telephony and communication features to existing applications. The components can be used in Windows®-based development environments that support ActiveX components. The Diva Component API provides functionalities including: extended call control, voice streaming and recording, change of media on existing calls, conferencing, call transfer, retrieve status information of the hardware, flexible tone detection, and answering machine detection.

Dialogic® Diva® API for .NET

The Diva API for .NET is based on the Dialogic® Diva® API. The functionality is exactly the same and where possible a one to one mapping is done. The Diva API for .NET documentation describes the architecture, the mapping of data types, and the modifications that have been done due to the .NET and especially the C# requirements.

Applications written for .NET may be based on two APIs provided by the Diva SDK: the Dialogic® Diva® Component API and the Dialogic® Diva® API for .NET. Both APIs use the feature rich Diva API to access the underlying Dialogic® communication platform. The Diva Component API allows for synchronous processing and is therefore often used by script-oriented applications. The decision as to which Dialogic® Diva® API can be used depends on the application requirements.

Extended CAPI 2.0

The Extended CAPI 2.0 is only available for Dialogic® Diva® Media Boards and provides Dialogic-specific extensions for CAPI 2.0. The extensions are fully CAPI 2.0 compatible, and thus can be used with CAPI 2.0 applications. The following Dialogic-specific CAPI extensions are available:

- Echo canceller support for voice applications: This extension allows the voice application to place an echo canceller unit in the front end of a connection to suppress acoustical echo and signal return. The Dialogic extension and the new CAPI standard for echo canceller are supported.
- Extension for fax paper formats and resolutions: This extension enables fax transmission and reception with an extended range of paper formats and resolutions.
- Tone detection and generation extension for DTMF facility: This extension enables fax and voice applications to detect in-band signals such as busy tone, to report events like modem CNG or fax flag detection, to detect human speech, to report the unidentified tones, and to report that no signal is present on the line.
- Extensions for modem configuration: This extension enables to specify certain modulation and protocol-related parameters. Modulations can be removed from the auto moding list or specific modulations can be selected. The results of the modulation and the protocol negotiation are signaled to the application.
- Generic tone generator and detector support for voice applications: This extension provides built-in generic tone detector and generator facilities.The generic tone services include sine generators with programmable frequency and amplitude modulation, function generators with programmable signal shape, frequency, and amplitude modulation, noise generators with programmable crest factor and amplitude modulation, single tone detection, and dual tone detection.

Descriptions of the Dialogic-specific CAPI 2.0 extensions are available under `SDK/DOC`. The CAPI 2.0 specification can be downloaded from the web site www.capi.org.

Note: If you are developing CAPI 2.0 applications based on Dialogic® Diva® for Windows® software, the CAPI 2.0 is part of the Diva API in the Dialogic® Diva® Configuration Manager.

Dialogic® Diva® Management API

The Diva Management API is only available for Dialogic® Diva® Media Boards and only applications based on the CAPI 2.0 may use this interface. Applications using the Dialogic® Diva® API, the Dialogic® Diva® Component API, or Dialogic® Diva® API for .NET should not use this API.

The Diva Management API offers a range of functions to retrieve status and statistics information:

- Retrieve status of lowest level ISDN access
- Retrieve active calls
- Generating statistic information for number of calls, etc.
- Notification of status changes

Applications using the Dialogic® Diva® API do not need to access the Diva Management API, because the Diva API already has built-in support for the Diva Management API.

The Diva Management API only registers information and executes functions within your system. You cannot use it for data transfer. For data transfer, you can use the Diva API or CAPI interface.

The status information provided by the Diva Management API is structured as a sort of "virtual" file space. It contains nodes (similar to directories) and values (similar to files), where each node and value is defined by its path and name. The information can be read out using Diva Management API functions. The DLL providing these functions is part of the Dialogic® Diva® for Windows® software.

It is planned that the functions provided by the Diva Management API will be extended periodically; however, it is also intended that the Diva Management API will remain backwards compatible such that applications based on an earlier Diva Management API can continue to be used with new versions.

Dialogic® communication platform-related information

The Dialogic® Diva® SDK uses the Dialogic® Diva® System Release software or the Dialogic® HMP software to communicate to the TDM or IP-based communication resources. The Diva-based software is automatically started at system start and configured via the Dialogic® Diva® Configuration Manager.

Dialogic® HMP software configuration and licensing

The Dialogic HMP software is started either automatically when the system starts or manually depending on the configuration in the Dialogic® HMP software's Configuration Manager (DCM). With the initialization of the Dialogic® Diva® API by the application, the Dialogic HMP software is initialized and configured. The configuration parameters are read from the file "dssdk.xml". Refer to the "Dialogic® HMP Software and Dialogic® Diva® SDK Installation and Configuration Guide" for details on those configuration parameters.

The Dialogic HMP software features are based on licenses, and there are various options that can be combined. Based on the available licenses, Dialogic® Diva® API interface functions may return *DivaErrorNotSupported* if a requested function is not licensed or no more licenses are available. The following section provides detailed information.

The Diva API supports voice, conferencing, and fax on the Dialogic HMP software. In addition, mixed conferences between Dialogic® Diva® Media Boards and the Dialogic HMP software are supported. Tromboning, called Line Interconnect on the Diva API, is also supported between Diva Media Boards and the Dialogic HMP software.

The following license options are validated during startup of the Diva API:

- IP call control / RTP G.711
- voice
- speech integration
- conferencing
- fax

The Diva SDK allocates resources for the duration of a call. If a call is initiated as a voice call, a voice resource is allocated and assigned to this call. This resource remains allocated even if the application switches to fax mode later.

IP call control / RTP G.711 resources

The amount of IP call control / RTP G.711 resources specifies the maximum number of channels. When started, the Dialogic® Diva® API creates the virtual line devices based on the configuration information that is specified in the configuration file. By default, only one line device is created for SIP-based communication using the available channels.

If line devices are configured and the amount of configured channels exceeds the licensed channels, the amount of channels for a line device or the amount of line devices may be limited. The application detects it in the information returned by *DivaGetNumLineDevices* and *DivaGetLineDeviceInfo*.

Voice and speech integration resources

The Dialogic® HMP software provides voice resources for supporting features such as streaming audio, detecting DTMF tones, and generating DTMF tones. The capabilities of a voice resource depend on the license. If only "voice" is licensed, this resource can either play or record, but not both at the same time. A "speech integration" resource can play and record in parallel, and it supports an echo canceller. Based on the available voice resources,

the Dialogic® Diva® API supports three different modes and the interface functions *DivaRecordVoiceFile* and *DivaSendVoiceFile* may behave differently. During system start, the Dialogic® Diva® SDK enumerates the Dialogic HMP software resources and selects one of the following three modes:

- Two voice resources per channel
- One speech integration voice resource per channel
- One voice resource per channel

Two voice resources per channel

This mode allows to play and record in parallel and it is entered if two voice resources are available for each licensed IP-channel. The behavior of the functions *DivaRecordVoiceFile* and *DivaSendVoiceFile* is the same as on Dialogic® Diva® Media Boards.

When a call is established, the received audio is signaled to the application via the event *DivaEventDataAvailable* and can be retrieved by the application via *DivaReceiveAudio*.

One speech integration voice resource per channel

This mode is entered if one speech integration license per IP-channel is available and it allows to play and to record in parallel. The functions *DivaRecordVoiceFile* and *DivaSendVoiceFile* behave like on Diva Media Boards. For received audio, the echo canceller can be enabled.

When a call is established, the received audio is signaled to the application via the event *DivaEventDataAvailable* and can be retrieved by the application via *DivaReceiveAudio*.

This mode allows for retrieving audio in small buffer sizes and is the base for bridging between TDM and IP-based calls.

One voice resource per channel

This mode is entered if voice resources are licensed but none of the previously described modes can be selected. This mode allows to play or to record but not at the same time. If a recording is active, any *DivaSendVoice* function will fail.

When a call is established, the received audio is not automatically signaled to the application via the event *DivaEventDataAvailable*. If the application requires this event, it must enable this via the function *DivaEnableRxData* and ensure that no play is active.

Conference resources

The Dialogic® Diva® SDK uses the HMP conference resources when an IP-based call is added to a conference. Note that creating a conference via *DivaCreateConference* will always succeed, even if no conference resource is licensed. The conference resource is allocated when an IP-based call is added to a conference using *DivaAddToConference*.

For each IP-based call that is added to a conference, one conference party resource is allocated and released when the call is disconnected or removed from the conference. An additional conference party resource is required if TDM and IP calls are bridged or if a play or record operation on the conference object is initiated. Note that once this additional resource is allocated, it remains at the conference object until the last IP-based call is removed from the conference or the conference object is released using *DivaDestroyConference*.

Applications may record from any conference member. Playing to an IP-based call that is part of a conference is not possible. In this case, the function *DivaSendVoiceFile* and the other sent voice-related functions will return *DivaErrorInvalidState*. The same is valid for a call that is line interconnected (tromboned) to another call.

Fax resources

The fax resources licensed for the Dialogic® HMP software support T.38 and clear channel fax, and the maximum supported speed is 14.400 bps. Fax resources are allocated when the application initiates a fax call or when the remote peer indicates a call as a fax call. For Dialogic HMP-based IP calls, the supported fax data format is TIFF. The following fax formats are supported:

- DivaFaxFormatTIFF_G3
- DivaFaxFormatTIFF_G4
- DivaFaxFormatColorJPEG

CHAPTER 2

Dialogic® Diva® API for .NET Overview

Diva API for .NET provides access to the Dialogic® Diva® API from .NET-based applications. The Diva API is an unmanaged interface and cannot be accessed directly by .NET applications. Therefore, Diva for .NET implements a wrapper that imports the functions from the Diva API DLL (DsSDK.DLL) and makes them available to .NET applications in the form of methods in a static class. The constants, enums, and structures from the DSAPI.H header file are also declared within this class.

Using the wrapper, a .NET-based application can use the Dialogic® Diva® API in an almost identical way to a native Win32 application written in C or C++.

Diva API for .NET also includes a small set of code samples that demonstrate the usage of the wrapper. These samples are available in source code. The samples are based on a framework that makes application development easy. The framework is also available in source.

Prerequisites

This manual assumes that the developer has knowledge on the .NET architecture and Microsoft® Visual Studio® 2003 or later.

Components and installation

The Dialogic® Diva® API for .NET requires the following files to be available on the target system.

- DivaServer.dll
- DsSDK.dll

Note: These files need to be on any target system, not only on the development platform. Therefore, the Dialogic® Diva® SDK does not install them in the Global Assembly Cache (GAC). The sample application contains a reference to the relative path of the DLLs. The DivaServer.dll contains a signature and is therefore prepared to be added to the GAC.

On a system using a Dialogic® Diva® communication platform the DsSDK.dll is typically installed to the %Windir%\System32 directory. Applications may deliver the DLLs used for developing the application and install them with the applications.

Documentation

The Dialogic® Diva® API for .NET is very similar to the Dialogic® Diva® API. Therefore, the documentation of the Diva API for .NET is based on the documentation of the Diva API.

This document describes the commonalities and differences between the two APIs. In most case, only the naming convention has been changed and the types have been converted to .NET types. For details on the functions, structures, enumerations, and constants, refer to the Dialogic® Diva® API documentation.

Object Browser

The Visual Studio® .NET object browser also shows the declaration of the functions and their parameters, the structures, enumerations, and the constants.

CHAPTER 3

Using the .NET Wrapper

The .NET wrapper is a thin layer that imports and adapts the functions provided by the Dialogic® Diva® API to make them available from managed code. Where the Common Language Runtime provides equivalent data types, these are used. Compound types (structures), enumerations and constants are re-declared, refer to Naming conventions for more information. Attribute tags of the .NET runtime are used to control the marshalling of the data.

Namespace

The namespace for the .NET wrapper is "DivaServer". The methods, structures, enumerations and constants are declared within a static class called "DSAPI".

Naming conventions

The Diva API as a native "C" interface uses the prefix "Diva" for the declarations to avoid any conflicts with other header files. Using a static class this name prefix is no longer needed and therefore the following naming conventions apply.

Note: The examples assume the "using DivaServer" statement is used.

Functions

The prefix "Diva" is removed. The functions are accessed using DSAPI.<function name>.

Diva API	DSAPI for .NET
DivaCloseCall(…)	DSAPI.CloseCall(…)
DivaSendVoiceFile(…)	DSAPI.SendVoiceFile(…)

Enumerations

For enumerations, not only the "Diva" prefix is removed, but also the enumeration name is removed from the members.

Diva API	Diva API for .NET

typedef enum
{
DivaProcessingGroupEvent = 1,
DivaProcessingGroupSending,
DivaProcessingGroupRecording,
} DivaProcessingGroup;
	public enum ProcessingGroup : uint
	{
	Event = 1,
	Sending,
	Recording,
	}

Usage example:

Param = DivaProcessingGroupEvent	Param = DSAPI.ProcessingGroup.Event

Structures

For structures, the "Diva" prefix is removed from the name of the structures. The members of a structure are used unchanged.

Unions

In general, the Common Language Runtime (CLR) does not support unions. There are commands and attributes to emulate the functionality of a union in the CLR, but there are some restrictions to the used types. Therefore, not all unions can be converted directly. For those functions that use not convertible unions, multiple implementations have been added to the DSAPI class using different parameter types. See Dialogic® Diva® API for .NET Functions on page 18 for more information on the functions.

Type conversion

The managed code of the CLR uses different types. The following tables show how common types are mapped and how the specific types defined by the Dialogic® Diva® API, e.g., handles, are mapped.

Common types

Diva API	Diva API for .NET
DWORD	UInt32 / uint DSAPI.ReturnCode *(function result)* (flag-type enumerations)
WORD	UInt16 / ushort
int	Int32 / int
unsigned char	Byte / byte
char* / const char* / char[ ]	String / string *(input)* StringBuilder *(output)*
char**	String[ ] / string[ ]
BOOL	Boolean / bool
PVOID	IntPtr
DWORD *	ref UInt32 *(input, in/out)* out UInt32 *(output)*
unsigned char* (buffer pointer)	byte[ ] *(input)* [Out] byte[ ] *(output)* IntPtr *(generic, manual marshalling)*

Dialogic® Diva® API-specific types

Diva API	Diva API for .NET
DivaAppHandle	IntPtr
DivaCallHandle	IntPtr
AppCallHandle	IntPtr
DivaMonitorHandle	IntPtr
AppMonitorHandle	IntPtr

Return codes of functions

The functions of the Dialogic® Diva® API return a DWORD value containing the result code defined in "DivaReturnCodes". The corresponding functions on the Dialogic® Diva® API for .NET return the same result codes but the type is DSAPI.ReturnCode.

CHAPTER 4

Dialogic® Diva® API for .NET Functions

This chapter provides information on functions of the DSAPI class that do not map identically to the Dialogic® Diva® API.

Registration

The Dialogic® Diva® API for .NET is an event driven API and requires a registration. There are several event modes available:

- Callback mode (different types available)
- Signal an application event object

For registration at the Diva API for .NET, different modes are available. Therefore, the functions have been split into different registration functions based on the type of event reporting.

Callback-based registrations

Applications using the callback method for event reporting register a delegate with the Dialogic® Diva® API for .NET. The delegate is called when an event is available. There are different types of delegates. Some directly signal the event with additional parameters, others just signal that an event is in the internal queue of the Diva API for .NET. Refer to the DSAPI.RegisterCallback variants for more information on the callback methods and the corresponding delegates.

DSAPI.RegisterCallback

With *DSAPI.RegisterCallback* the application registers with the Dialogic® Diva® API for .NET using a delegate of type EventCallback.

```
ReturnCode      RegisterCallback (      out IntPr           pHandle,
                                        EventCallback       Callback,
                                        UInt32              MaxConn,
                                        UInt32              RxBuffers,
                                        UInt32              MaxBufferSize )
```

Parameters

pHandle

[out] This parameter is a pointer to a location that receives the handle for further access to the Diva API for .NET.

Callback

[in] The parameter specifies the delegate to be called when an event is available. Refer to DSAPI.EventCallback for more information.

MaxConn

[in] This parameter specifies the maximum number of connections to be used by the application. If this parameter is set to zero, a maximum of one connection per available physical channel can be used.

RxBuffers

[in] This parameter specifies the number of data blocks that should be reserved for each connection in receive direction.

MaxBufferSize

[in] This parameter specifies the maximum buffer size to be used. See remarks below.

Return values

If the function succeeds, the return value is *DivaSuccess* (0). Other possible return values are *DSAPI.ReturnCode.ErrorLineDevice*, *DSAPI.ReturnCode.ErrorInvalidHandle*, and *DSAPI.ReturnCode.ErrorInvalidParameter*.

Remarks

This function must be called by the application before any connection-oriented function can be called. The application registers the parameters to be used for the amount of concurrent connections and the data buffer management.

The *MaxBufferSize* and *RxBuffers* parameters are valid for connections done on this registration at the Dialogic® Diva® API. The defaults are selected to have an optimal situation for fax and voice connections. The amount of buffers should be between 4 and 8. This amount of buffers is used for receiving data and also for sending data. The *MaxBufferSize* depends on the application requirements. If only fax or data calls are handled, the maximum of 2048 should be used. For voice applications, the delay may be relevant. If the application is delay sensitive, a buffer size or 256 should not be exceeded. If the application is not delay sensitive, a 1024 or even 2048 is recommended. The buffer size should not be below 128.

The delegate passed to *DSAPI.RegisterCallback* must be of type *DSAPI.EventCallback*. The event and the corresponding event-specific parameters are passed to the delegate. For more information, see DSAPI.EventCallback.

DSAPI.RegisterCallbackEx

With *DSAPI.RegisterCallbackEx* the application registers with the Dialogic® Diva® API for .NET using a delegate of type EventCallbackEx.

```
ReturnCode    RegisterCallbackEx (    out IntPr          pHandle,
                                      EventCallbackEx    Callback,
                                      IntPr              Context
                                      UInt32             MaxConn,
                                      UInt32             RxBuffers,
                                      UInt32             MaxBufferSize )
```

pHandle

[out] This parameter is a pointer to a location that receives the handle for further access to the Diva API for .NET.

Callback

[in] The parameter specifies the delegate to be called when an event is available. Refer to DSAPI.EventCallbackEx for more information.

Context

[in] The parameter specifies the application context. The context is not interpreted by the Dialogic® Diva® SDK and passed unchanged to the delegate when an event is signaled.

MaxConn

[in] This parameter specifies the maximum number of connections to be used by the application. If this parameter is set to zero, a maximum of one connection per available physical channel can be used.

RxBuffers

[in] This parameter specifies the number of data blocks that should be reserved for each connection in receive direction.

MaxBufferSize

[in] This parameter specifies the maximum buffer size to be used. See remarks below.

Return values

If the function succeeds, the return value is *DivaSuccess* (0). Other possible return values are *DSAPI.ReturnCode.ErrorLineDevice*, *DSAPI.ReturnCode.ErrorInvalidHandle*, and *DSAPI.ReturnCode.ErrorInvalidParameter*.

Remarks

This function must be called by the application before any connection-oriented function can be called. The application registers the parameters to be used for the amount of concurrent connections and the data buffer management.

The *MaxBufferSize* and *RxBuffers* parameters are valid for connections done on this registration at the Dialogic® Diva® API. The defaults are selected to have an optimal situation for fax and voice connections. The amount of buffers should be between 4 and 8. This amount of buffers is used for receiving data and also for sending data. The *MaxBufferSize* depends on the application requirements. If only fax or data calls are handled, the maximum of 2048 should be used. For voice applications, the delay may be relevant. If the application is delay sensitive, a buffer size or 256 should not be exceeded. If the application is not delay sensitive, a 1024 or even 2048 is recommended. The buffer size should not be below 128.

The delegate passed to *DSAPI.RegisterCallbackEx* must be of type *DSAPI.EventCallbackEx*. The application context and the event with the corresponding event-specific parameters are passed to the delegate. For more information, see DSAPI.EventCallbackEx.

DSAPI.RegisterCallbackSignal

With *DSAPI.RegisterCallbackSignal* the application registers with the Dialogic® Diva® API for .NET using a delegate of type EventCallbackSignal.

```
ReturnCode    RegisterCallbackSignal (    out IntPr              pHandle,
                                          EventCallbackSignal    Callback,
                                          IntPr                  Context
                                          UInt32                 MaxConn,
                                          UInt32                 RxBuffers,
                                          UInt32                 MaxBufferSize )
```

pHandle

[out] This parameter is a pointer to a location that receives the handle for further access to the Dialogic® Diva® API for .NET.

Callback

[in] The parameter specifies the delegate to be called when an event is available. Refer to DSAPI.EventCallbackSignal for more information.

Context

[in] The parameter specifies the application context. The context is not interpreted by the Dialogic® Diva® SDK and passed unchanged to the delegate when an event is signaled.

MaxConn

[in] This parameter specifies the maximum number of connections to be used by the application. If this parameter is set to zero, a maximum of one connection per available physical channel can be used.

RxBuffers

[in] This parameter specifies the number of data blocks that should be reserved for each connection in receive direction.

MaxBufferSize

[in] This parameter specifies the maximum buffer size to be used. See remarks below.

Return values

If the function succeeds, the return value is *DivaSuccess* (0). Other possible return values are *DSAPI.ReturnCode.ErrorLineDevice*, *DSAPI.ReturnCode.ErrorInvalidHandle*, and *DSAPI.ReturnCode.ErrorInvalidParameter*.

Remarks

This function must be called by the application before any connection-oriented function can be called. The application registers the parameters to be used for the amount of concurrent connections and the data buffer management.

The *MaxBufferSize* and *RxBuffers* parameters are valid for connections done on this registration at the Dialogic® Diva® API. The defaults are selected to have an optimal situation for fax and voice connections. The amount of buffers should be between 4 and 8. This amount of buffers is used for receiving data and also for sending data. The *MaxBufferSize* depends on the application requirements. If only fax or data calls are handled, the maximum of 2048 should be used. For voice applications, the delay may be relevant. If the application is delay sensitive, a buffer size or 256 should not be exceeded. If the application is not delay sensitive, a 1024 or even 2048 is recommended. The buffer size should not be below 128.

The delegate passed to *DSAPI.RegisterCallbackSignal* must be of type *DSAPI.EventCallbackSignal*. The application context is passed to the delegate. The event itself remains in the queue of the Diva API for .NET. The application must retrieve the event using DSAPI.GetEvent. For more information, see DSAPI.EventCallbackSignal.

DSAPI.EventCallback

Applications registering via *DSAPI.RegisterCallback* must implement a delegate of type *DSAPI.EventCallback*.

```
public delegate    void EventCallback (    IntPr     AppHandle,
                                           Event     EventType,
                                           IntPtr    EventParam1,
                                           IntPtr    EventParam2 );
```

Parameters

AppHandle

[in] This parameter specifies the application instance. It is the handle returned by the Dialogic® Diva® API for .NET during registration via *DSAPI.RegisterCallback*.

EventType

[in] The *EventType* parameter specifies the event that causes the call to this function. For more information, see the list of events in the Dialogic® Diva® API documentation.

EventParam1

[in] This parameter is event-specific. For call-related events except for signaling a new call, this parameter is usually the application handle passed into *DSAPI.Connect...*, *DSAPI.Answer...*, or *DSAPI.CreateCall*.

EventParam2

[in] This parameter is event-specific.

Return values

None.

Remarks

The delegate is called as soon as a new event is available. The delegate is called on the thread context of the Diva API for .NET. The delegate is not re-entered.

DSAPI.EventCallbackEx

Applications registering via *DSAPI.RegisterCallbackEx* must implement a delegate of type *DSAPI.EventCallbackEx*.

```
public delegate    void EventCallbackEx (    IntPr      Context,
                                             Event      EventType,
                                             IntPtr     EventParam1,
                                             IntPtr     EventParam2 );
```

Parameters

Context

[in] This parameter has been provided as parameter *Context* by the application with the call to *DSAPI.RegisterCallbackEx*.

EventType

[in] The *EventType* parameter specifies the event that causes the call to this function. For more information, see the list of events in the Dialogic® Diva® API documentation.

EventParam1

[in] This parameter is event-specific. For call-related events except for signaling a new call, this parameter is usually the application handle passed into *DSAPI.Connect...*, *DSAPI.Answer...*, or *DSAPI.CreateCall*.

EventParam2

[in] This parameter is event-specific.

Return values

None.

Remarks

The delegate is called as soon as a new event is available. The delegate is called on the thread context of the Dialogic® Diva® API for .NET. It is ensured that the delegate is not re-entered.

DSAPI.EventCallbackSignal

Applications registering via *DSAPI.RegisterCallbackSignal* must implement a delegate of type *DSAPI.EventCallbackSignal*.

```
public delegate    void EventCallbackSignal (    IntPr    Context );
```

Parameters

Context

[in] This parameter has been provided as parameter *Context* by the application with the call to *DSAPI.RegisterCallbackSignal*.

Return values

None.

Remarks

The delegate is called as soon as a new event is available. The delegate is called on the thread context of the Dialogic® Diva® API for .NET. It is ensured that the delegate is not reentered. The application must retrieve the event by calling *DSAPI.GetEvent*. This can be done from the delegate or later depending on the application.

Event-based registration

Applications using the event-based registration, register an event object with the Dialogic® Diva® API for .NET. This event object is signalled whenever there are events available. Once the event object is signaled, the application must retrieve events by calling DSAPI.GetEvent() in a loop until the queued events have been processed.

DSAPI.RegisterEvent using event handle

The *DSAPI.RegisterEvent* function registers with the Dialogic® Diva® API for .NET using an event object and sets global parameters.

```
ReturnCode      RegisterEvent (     out IntPr       pHandle,
                                    IntPr           EventObject,
                                    UInt32          MaxConn,
                                    UInt32          RxBuffers,
                                    UInt32          MaxBufferSize );
```

Parameters

pHandle

[out] This parameter is a pointer to a location that receives the handle for the further access to the Diva API for .NET.

EventObject

[in] The parameter specifies the event object to be signaled when an event is available. For details, refer to the Remarks section.

MaxConn

[in] This parameter specifies the maximum number of connections to be used by the application. If this parameter is set to zero, a maximum of one connection per available physical channel can be used.

RxBuffers

[in] This parameter specifies the number of data blocks that should be reserved for each connection in receive direction.

MaxBufferSize

[in] This parameter specifies the maximum buffer size to be used. See remarks below.

Return values

If the function succeeds, the return value is *DivaSuccess* (0). Other possible return values are *DSAPI.ReturnCode.ErrorLineDevice*, *DSAPI.ReturnCode.ErrorInvalidHandle*, and *DSAPI.ReturnCode.ErrorInvalidParameter*.

Remarks

This function must be called by the application before any connection-oriented function can be called. The application registers the parameters to be used for the amount of concurrent connections and the data buffer management.

The *MaxBufferSize* and *RxBuffers* parameters are valid for connections done on this registration at the Dialogic® Diva® API. The defaults are selected to have an optimal situation for fax and voice connections. The amount of buffers should be between 4 and 8. This amount of buffers is used for receiving data and also for sending data. The *MaxBufferSize* depends on the application requirements. If only fax or data calls are handled, the maximum of 2048 should be used. For voice applications, the delay may be relevant. If the application is delay sensitive, a buffer size or 256 should not be exceeded. If the application is not delay sensitive, a 1024 or even 2048 is recommended. The buffer size should not be below 128.

The event object may be either a raw Win32 event object, created by native Win32 functions imported from kernel32.dll, or the equivalent provided by the .NET framework, i.e. System.Threading.AutoResetEvent or System.Threading.ManualResetEvent. The objects of the .NET framework have a public property named "Handle" that returns the native Win32 handle of the underlying event object as .NET Type "IntPtr".

DSAPI.RegisterEvent using System.Threading.AutoResetEvent

The *DSAPI.RegisterEvent* function registers with the Dialogic® Diva® API for .NET using an event object and sets global parameters.

```
ReturnCode    RegisterEvent (    out IntPr                          pHandle,
                                 System.Threading.AutoResetEvent    EventObject,
                                 UInt32                             MaxConn,
                                 UInt32                             RxBuffers,
                                 UInt32                             MaxBufferSize );
```

Parameters

pHandle

[out] This parameter is a pointer to a location that receives the handle for further access to the Diva API for .NET.

EventObject

[in] The parameter specifies the event object of type *System.Threading.AutoResetEvent*. The event will be signaled when an event is available.

MaxConn

[in] This parameter specifies the maximum number of connections to be used by the application. If this parameter is set to zero, a maximum of one connection per available physical channel can be used.

RxBuffers

[in] This parameter specifies the number of data blocks that should be reserved for each connection in receive direction.

MaxBufferSize

[in] This parameter specifies the maximum buffer size to be used. See remarks below.

Return values

If the function succeeds, the return value is *DivaSuccess* (0). Other possible return values are *DSAPI.ReturnCode.ErrorLineDevice*, *DSAPI.ReturnCode.ErrorInvalidHandle*, and *DSAPI.ReturnCode.ErrorInvalidParameter*.

Remarks

This function must be called by the application before any connection-oriented function can be called. The application registers the parameters to be used for the amount of concurrent connections and the data buffer management.

The *MaxBufferSize* and *RxBuffers* parameters are valid for connections done on this registration at the Dialogic® Diva® API. The defaults are selected to have an optimal situation for fax and voice connections. The amount of buffers should be between 4 and 8. This amount of buffers is used for receiving data and also for sending data. The *MaxBufferSize* depends on the application requirements. If only fax or data calls are handled, the maximum of 2048 should be used. For voice applications, the delay may be relevant. If the application is delay sensitive, a buffer size or 256 should not be exceeded. If the application is not delay sensitive, a 1024 or even 2048 is recommended. The buffer size should not be below 128.

Native registration function

In addition to the registration methods, specific to the event mode the Dialogic® Diva® API for .NET provides a method that allows specifying the native parameter for registrations. This function is similar to the function *DivaRegister* of the Dialogic® Diva® API. The function declaration is shown below. For parameter description, refer to the documentation of *DivaRegister*.

```
ReturnCode      Register (      out IntPr       pHandle,
                                EventMode       EventMode,
                                IntPtr          EventModeSpecific1,
                                IntPtr          EventModeSpecific2,
                                UInt32          MaxConn,
                                UInt32          RxBuffers,
                                UInt32          MaxBufferSize )
```

Call Properties

The Dialogic® Diva® API as well as the Dialogic® Diva® API for .NET support a long list of call properties. The call properties have various types, some are Boolean other exchange a string or even binary data. For each required data type, an implementation of *DSAPI.SetCallProperties* and *DSAPI.GetCallProperties* is available.

The call properties are described in detail in the Dialogic® Diva® API documentation. Below are the declarations for *DSAPI.SetCallProperties* and *DSAPI.GetCallProperties*. Each declaration has a unique number. The table of call properties lists for each property the .NET type and the number of the corresponding *DSAPI.SetCallProperties* or *DSAPI.GetCallProperties* variant.

The framework for the samples contains the object CallBase. This shows for each property how it can be retrieved.

Number	Declaration			
1	ReturnCode	SetCallProperties (	IntPtr CallPropertyType UInt32	hCall, Type, Value)
2	ReturnCode	SetCallProperties (	IntPtr CallPropertyType String	hCall, Type, Value)
3	ReturnCode	SetCallProperties (	IntPtr CallPropertyType ref NumberInformation UInt32	hCall, Type, Value, Size)
4	ReturnCode	SetCallProperties (	IntPtr CallPropertyType ref BinaryData	hCall, Type, Value)
5	ReturnCode	GetCallProperties (	IntPtr CallPropertyType IntPtr UInt32	hCall, Type, Value, Size)
6	ReturnCode	GetCallProperties (	IntPtr CallPropertyType out UInt32	hCall, Type, Value)
7	ReturnCode	GetCallProperties (	IntPtr CallPropertyType System.Text.StringBuilder UInt32	hCall, Type, Value, Size)
8	ReturnCode	GetCallProperties (	IntPtr CallPropertyType ref NumberInformation UInt32	hCall, Type, Value, Size)
9	ReturnCode	GetCallProperties (	IntPtr CallPropertyType ref BinaryData UInt32	hCall, Type, Value, Size)
10	ReturnCode	GetCallProperties (	IntPtr CallPropertyType out SetupMessate UInt32	hCall, Type, Value, Size)

Property	.NET Type	Get Function	Set Function
CallType	UInt32	6	1
LineDevice	UInt32	6	1
SignaledService	UInt32	6	1
BearerCapabilities	BinaryData	9	4
CalledNumber	System.Text.StringBuilder(Get)	7	
CallingNumber	String(Set) System.Text.StringBuilder(Get)	7	2
CalledNumberParams	NumberInformation	8	3
CallingNumberParams	NumberInformation	8	3
RedirectingNumber	System.Text.StringBuilder(Get)	7	
RedirectedNumber	System.Text.StringBuilder(Get)	7	
TxSpeed	UInt32	6	
RxSpeed	UInt32	6	
DiscReason	UInt32	6	
SignaledLineDiscReason	UInt32	6	
RejectReason	UInt32		1
VoiceEchoCanceler	UInt32		1
VoiceEarlyDataChannel	UInt32		1
VoiceRecordSilenceTimeout	UInt32		1
SecondCallingNumber	System.Text.StringBuilder(Get)	7	
SecondCallingNumberParams	NumberInformation	8	
CallingName	String(Set) System.Text.StringBuilder(Get)	7	2
ConnectedName	System.Text.StringBuilder(Get)	7	
CallingSubAddress	String(Set) System.Text.StringBuilder(Get)	7	2
CalledSubAddress	String(Set) System.Text.StringBuilder(Get)	7	2
OriginalCalledNumber	System.Text.StringBuilder(Get)	7	
ConnectedNumber	System.Text.StringBuilder(Get)	7	
CalledName	System.Text.StringBuilder(Get)	7	
DataChannel	UInt32	6	1
DisconnectReason	UInt32		1
DisconnectCause	UInt32		1
RedirectionNumber	System.Text.StringBuilder(Get)	7	
VoiceRecordStartTones	String(Set)		2
VoiceDTMF_SendDuration	UInt32		1
VoiceDTMF_SendPause	UInt32		1
VoiceDTMF_DetectDuration	UInt32		1
VoiceDTMF_DetectPause	UInt32		1
VoiceRemoveDTMFFromStream	UInt32		1
VoiceEarlyDataDiscOnInfo	UInt32		1

Property	.NET Type	Get Function	Set Function
EchoCancellerEnableNLP	UInt32		1
EchoCancellerAutoDisable1	UInt32		1
EchoCancellerAutoDisable2	UInt32		1
EchoCancellerTailLength	UInt32		1
EchoCancellerPreDelay	UInt32		1
EnableDTMFTrailingEdge	UInt32		1
FaxLocalId	String(Set)		2
FaxHeadline	String(Set)		2
FaxRemoteId	System.Text.StringBuilder(Get)	7	
FaxPages	UInt32		1
FaxMaxSpeed	UInt32		1
FaxHighResolution	UInt32		1
FaxEnablePolling	UInt32		1
FaxReverseSession	UInt32		1
FaxMultiDocument	UInt32		1
FaxDisableECM	UInt32		1
FaxDisableMR	UInt32		1
FaxDisableMMR	UInt32		1
FaxPageQuality	UInt32	6	
FaxPageEndInfo	UInt32	6	
FaxRemoteFeatures	BinaryData	9	
FaxRemoteMaxHorzRes	UInt32	6	
FaxRemoteMaxVertRes	UInt32	6	
FaxRemoteMaxSpeed	UInt32	6	
FaxRemoteNSF	BinaryData	7	
FaxLocalNSF	BinaryData		4
FaxEnableColor	UInt32		1
FaxColorSelected	UInt32	6	
EnableInterrupt	UInt32		1
RequestInterrupt	UInt32		1
FaxProcedureInterrupt	UInt32	6	
FaxEnableSecurity	UInt32		1
FaxRemoteSupportsSubaddr	UInt32	6	
FaxRemoteSupportsPassword	UInt32	6	
FaxSignalSubAddress	String(Set)		2
FaxSignalPassword	String(Set)		2
FaxRemoteSubAddress	System.Text.StringBuilder(Get)	7	
FaxRemotePassword	System.Text.StringBuilder(Get)	7	
FaxDisableFileBuffering	UInt32		1
FaxUseTextForSending	UInt32		1

Property	.NET Type	Get Function	Set Function
FaxAllowDocumentStretching	UInt32		1
FaxRemoteScanLineLength	UInt32		1
FaxStoreMode	UInt32		1
FaxPassDataOnNextPageStart	UInt32		1
FaxStartPage	UInt32		1
MaximumSpeed	UInt32		1
DataBits	UInt32	6	1
StopBits	UInt32	6	1
Parity	UInt32	6	1
DisableCompression	UInt32		1
DisableV42	UInt32		1
DisableMNP	UInt32		1
ForceReliable	UInt32		1
DisableRetrain	UInt32		1
ModulationClass	UInt32		1
NegotiatedV42V42bis	UInt32	6	
NegotiatedMNP4MNP5	UInt32	6	
NegotiatedTransparent	UInt32	6	
NegotiatedCompression	UInt32	6	
DCD	UInt32	6	
CTS	UInt32	6	
ConnectedNorm	UInt32	6	
RoundTripDelay	UInt32	6	
GuardTone	UInt32		1
ModemLeasedLine	UInt32		1
Modem4WireOption	UInt32		1
DisableDiscOnBusyTone	UInt32		1
DisableCallingTone	UInt32		1
DisableAnswerTone	UInt32		1
EnableDialToneDetect	UInt32		1
DisableStepUp	UInt32		1
DisableStepDown	UInt32		1
DisableSpiltSpeed	UInt32		1
DisableTrellisCoding	UInt32		1
DisableFlushTimer	UInt32		1
EnableEmptyFrames	UInt32		1
EnableMultimoding	UInt32		1
BypassControl	UInt32		1
DisableModulationV21	UInt32		1
DisableModulationV22	UInt32		1

Property	.NET Type	Get Function	Set Function
DisableModulationV22bis	UInt32		1
DisableModulationV23	UInt32		1
DisableModulationV32	UInt32		1
DisableModulationV32bis	UInt32		1
DisableModulationV34	UInt32		1
DisableModulationV90DPCM	UInt32		1
DisableModulationBell103	UInt32		1
DisableModulationBell212A	UInt32		1
DisableModulationAllFS	UInt32		1
DisableModulationK56Flex	UInt32		1
DisableModulationX2	UInt32		1
DisableV42Detection	UInt32		1
EnableModulationV29FDX	UInt32		1
EnableModulationV33	UInt32		1
EnableModulationV90APCM	UInt32		1
EnableModulationV22FS	UInt32		1
EnableModulationV29FS	UInt32		1
EnableModulationV23_1	UInt32		1
EnableModulationV23_2	UInt32		1
MinimumTxSpeed	UInt32		1
MaximumTxSpeed	UInt32		1
MinimumRxSpeed	UInt32		1
MaximumRxSpeed	UInt32		1
TxLevelAdjust	UInt32		1
DisableV34TxLevelReduction	UInt32		1
DisableV34PreCoding	UInt32		1
DisableV34PreEmphasis	UInt32		1
DisableV34Shaping	UInt32		1
DisableV34NonLinearEncoding	UInt32		1
DisableV34ManualReduction	UInt32		1
DisableV34Training16Point	UInt32		1
DisableV34SymbolRate2400	UInt32		1
DisableV34SymbolRate2743	UInt32		1
DisableV34SymbolRate2800	UInt32		1
DisableV34SymbolRate3000	UInt32		1
DisableV34SymbolRate3200	UInt32		1
DisableV34SymbolRate3429	UInt32		1
ForceReliableIfV34	UInt32		1
DisableSDLC	UInt32		1
DisableReliableIf1200	UInt32		1

Property	.NET Type	Get Function	Set Function
BufferDuringV42Detection	UInt32		1
DisableV42SelectivReject	UInt32		1
DisableMNP3	UInt32		1
DisableMNP4	UInt32		1
DisableMNP10	UInt32		1
TransparentModeIfV22bis	UInt32		1
TransparentModeIfV32bis	UInt32		1
BreakMode	UInt32		1
ModemEarlyConnect	UInt32		1
ModemPassIndication	UInt32		1
SDLCLinkAddress	UInt32		1
SDLCModuloMode	UInt32		1
SDLCWindowSize	UInt32		1
SDLCXID	UInt32		1
SDLCReverseEstablishment	UInt32		1
V18Selected	UInt32		1
V18ProbingSequence	UInt32		1
V18CountryProbingSequence	UInt32		1
V18ProbingMessage	UInt32		1
V18ReinitializeOnSilence	UInt32		1
V18RevertToAnswerMode	UInt32		1
V18DisconnectOnBusy	UInt32		1
V18AutomodingMonitorMode	UInt32		1
V18TextProbingForCarrierMode	UInt32		1
V18TXPSpaceParityInOrigMode	UInt32		1
V18EnableV18OriginationMode	UInt32		1
V18EnableV18AnswerMode	UInt32		1
V18EnableV21OriginationMode	UInt32		1
V18EnableV21AnswerMode	UInt32		1
V18EnableBell103OrigMode	UInt32		1
V18EnableBell103AnswerMode	UInt32		1
V18EnableV23OriginationMode	UInt32		1
V18EnableV23AnswerMode	UInt32		1
V18EnableEDTMode	UInt32		1
V18EnableBAUDOT45Mode	UInt32		1
V18EnableBAUDOT47Mode	UInt32		1
V18EnableBAUDOT50Mode	UInt32		1
V18EnableDTMFMode	UInt32		1
V18TransmitLevel	UInt32		1
V18AsyncFormatV21	UInt32		1

Property	.NET Type	Get Function	Set Function
V18AsyncFormatV23	UInt32		1
V18AsyncFormatBell103	UInt32		1
V18AsyncFormatEDT	UInt32		1
V18AsyncFormatBAUDOT	UInt32		1
V18TimerTcTimeout	UInt32		1
V18TimerTmTimeout	UInt32		1
V18CleanCarrierTime	UInt32		1
V18EchoSupressTime	UInt32		1
EnableModulationV22bisFS	UInt32		1
SDLCInitiateFastEstablishment	UInt32		1
B1Protocol	UInt32		1
B2Protocol	UInt32		1
B3Protocol	UInt32		1
B1Configuration	BinaryData		4
B2Configuration	BinaryData		4
B3Configuration	BinaryData		4
LLC	BinaryData	9	4
HLC	BinaryData	9	4
B_ChannelInfo	BinaryData		4
KeypadFacility	BinaryData	9	4
UserUserInfo	BinaryData	9	4
FacilityDataArray	BinaryData	9	4
DisplayInfo	BinaryData	9	
TotalChargeUnits	UInt32	6	
SpecialInfoElement	BinaryData	9	4
ChannelInfoElement	BinaryData	9	
ProgressIndElement	BinaryData	9	
SetupMessage	DivaSetupMessage	10	
GlobalConfiguration	UInt32		1
ReverseDataChannelConnect	UInt32		1
CauseInfoElement	BinaryData	9	
X25_NCPI	UInt32		1
X25_CalledAddress	UInt32		1
X25_CallingAddress	UInt32		1
AutoDetectMode	UInt32		1
AutoDetectX75ForceX25	UInt32		1
AutoDetectMaxFrames	UInt32		1
AutoDetectMaxSeconds	UInt32		1
UseSameChannelForTransfer	UInt32		1
NoAnswerTimeout	UInt32		1

Property	.NET Type	Get Function	Set Function
ConnectTimeout	UInt32		1
NoHoldBeforeTransfer	UInt32		1
UseExternalEquipment	UInt32		1
CallTimeStats	IntPtr	5	

Line Device Configuration

DSAPI.GetLineDeviceConfiguration

DSAPI.GetLineDeviceConfiguration retrieves information about the line device configuration.

ReturnCode	GetLineDeviceConfiguration (	UInt32 DeviceConfigType out UInt32 UInt32	LineDeviceId, Type, Value, ValueSize)
ReturnCode	GetLineDeviceConfiguration (	UInt32 DeviceConfigType System.Text.StringBuilder UInt32	LineDeviceId, Type, Value, ValueSize)

Parameters

LineDeviceId

[in] This parameter identifies the line device by an index starting with one.

Type

[in] This parameter specifies which configuration parameter should be read. For valid types, see the Remarks section.

Value (UInt32)

[out] This parameter is of type UInt32 and receives the value of the requested type.

Value (System.Text.StringBuilder)

[out] This parameter is of type System.Text.StringBuilder and receives the value of the requested type.

ValueSize

[out] This parameter specifies the length in bytes of Value. If the value is a UInt32 it is set to the size of the Uint32 (4 bytes). In case of System.Text.StringBuilder the size allocated, must be specified here.

Return values

If the function succeeds, the return value is *DSAPI.ReturnCode.Success* (0). Possible other return values are *DSAPI.ReturnCode.ErrorInvalidParameter*, *DSAPI.ReturnCode.DataSize*, and *DSAPI.ReturnCode.LineDevice*.

Remarks

The function reads the specified configuration property of the specified line device. The available configuration parameters are:

DSAPI.DeviceConfigType.	Data Type
DCT_SwitchType	UInt32
DCT_PBXName	UInt32
DCT_DDIEnabled	UInt32
DCT_Layer2Mode	UInt32
DCT_NTMode	UInt32
DCT_NumberCollectLength	UInt32
DCT_AutoSpid	UInt32
DCT_SPID1	System.Text.StringBuilder
DCT_SPID2	System.Text.StringBuilder
DCT_DirectoryNumber1	System.Text.StringBuilder
DCT_DirectoryNumber2	System.Text.StringBuilder

Line Device Status

DSAPI.GetLineDeviceStatus

DSAPI.GetLineDeviceStatus retrieves information about the line device status.

ReturnCode	GetLineDeviceStatus (	UInt32 DeviceConfigType out UInt32 UInt32	LineDeviceId, Type, Value, ValueSize)
ReturnCode	GetLineDeviceStatus (	UInt32 DeviceConfigType out DSPStateArray UInt32	LineDeviceId, Type, Value, ValueSize)
ReturnCode	GetLineDeviceStatus (	UInt32 DeviceConfigType ref PotsLineState UInt32	LineDeviceId, Type, Value, ValueSize)

Parameters

LineDeviceId

[in] This parameter identifies the line device by an index starting with one.

Type

[in] This parameter specifies which status parameter should be read. For valid types, see the Remarks section.

Value (UInt32)

[out] This parameter is of type UInt32 and receives the value of the requested type.

Value (DSAPI.DSPStateArray)

[out] This parameter is of type DSAPI.DSPStateArray and receives the states of the DSPs.

Value (PotsLineState)

[in out] This parameter is of type PotsLineState and specifies the line for which the status is requested. On return it receives the status of the line.

ValueSize

[out] This parameter specifies the length in bytes of Value. In case value is a UInt32 this is set to the size of the Uint32 (4 bytes). In case of DSAPI. DSPStateArray the size of the available buffer must be specified here.

Return values

If the function succeeds, the return value is *DSAPI.ReturnCode.Success* (0). Possible other return values are *DSAPI.ReturnCode.ErrorInvalidParameter*, *DSAPI.ReturnCode.DataSize*, and *DSAPI.ReturnCode.LineDevice*.

Remarks

The function reads the specified status property of the specified line device. The available status parameters are:

DSAPI.DeviceStatusType.	Data Type
Layer1Status	UInt32 (DSAPI. Layer1Status)
PotsLineStatus	DSAPI.PotsLineStatus
Layer2Status	UInt32 (DSAPI.Layer2Status)
RedAlarm	UInt32
BlueAlarm	UInt32
YellowAlarm	UInt32

DSAPI.DeviceStatusType.	Data Type
ActiveInConnections	UInt32
ActiveOutConnections	UInt32
TotalDSPs	UInt32
OutOfServiceDSPs	UInt32
DSPStates	DSAPI.DSPStateArray

Conference Properties

DSAPI.ConferenceGetProperties

DSAPI.ConferenceGetProperties retrieves information about the conference members.

```
ReturnCode    ConferenceGetProperties (    IntPtr                          hdConference,
                                           ConferencePropertyType          PropertyType,
                                           out UInt32                      PropertyValue,
                                           UInt32                          PropertyValueSize,
                                           out UInt32                      pPropertyValueSizeUsed )

ReturnCode    ConferenceGetProperties (    IntPtr                          hdConference,
                                           ConferencePropertyType          PropertyType,
                                           ref ConferenceMemberRights      PropertyValue,
                                           UInt32                          PropertyValueSize,
                                           out UInt32                      pPropertyValueSizeUsed )

ReturnCode    ConferenceGetProperties (    IntPtr                          hdConference,
                                           ConferencePropertyType          PropertyType,
                                           [Out] ConferenceMemberInfo [ ]  PropertyValue,
                                           UInt32                          PropertyValueSize,
                                           out UInt32                      pPropertyValueSizeUsed )
```

Parameters

hdConference

[in] The *hdConference* parameter identifies the conference previously created by *DivaCreateConference*.

PropertyType

[in] The *PropertyType* parameter specifies the property to be retrieved. For more information on the available properties for reading, see the Remarks section.

PropertyValue (UInt32)

[out] This parameter is of type UInt32 and receives the value of the requested type.

PropertyValue (ConferenceMemberRights)

[out] This parameter is of type ConferenceMemberRights and receives the rights of the members.

PropertyValue (ConferenceMemberInfo)

[out] This parameter is of type ConferenceMemberInfo and receives the information of the members.

PropertyValueSize

[in] The *PropertyValueSize* parameter specifies the size of the buffer provided by the application. In case value is a UInt32 this is set to the size of the Uint32 (4 bytes). In case of the other objects the size of operator of the .NET marshaler must be used.

pPropertyValueSizeUsed

[out] The *PropertyValueSizeUsed* parameter points to a location that receives the amount or bytes written to PropertyValue.

Return values

If the function succeeds, the return value is *DSAPI.ReturnCode.Success* (0). If the buffer for the property value is not large enough, *DSAPI.ReturnCode.OutOfMemory* is returned. Other possible return values are *DSAPI.ReturnCode.ErrorInvalidParameter* and *DSAPI.ReturnCode.InvalidHandle*.

Remarks

The function reads the specified conference property. The available conference properties and the corresponding data types are:

DSAPI.ConferencePropertyType.	Data Type
MaxMembers	UInt32
Options	UInt32 (DSAPI.ConferenceOptions)
DefRights	UInt32 (DSAPI.ConferenceRights)
MemberRights	DSAPI.ConferenceMemberRights
Supervisor	N/A for get function
NumMembers	UInt32
Members	DSAPI.ConferenceMemberInfo
NumTalkers	UInt32
Talkers	DSAPI.ConferenceMemberInfo

DSAPI.ConferenceSetProperties

DSAPI.ConferenceSetProperties modifies the state or attributes of the conference or conference members.

```
ReturnCode    ConferenceGetProperties (    IntPtr                      hdConference,
                                           ConferencePropertyType      PropertyType,
                                           ref UInt32                  PropertyValue )

ReturnCode    ConferenceSetProperties (    IntPtr                      hdConference,
                                           ConferencePropertyType      PropertyType,
                                           ref ConferenceMemberRights  PropertyValue )

ReturnCode    ConferenceSetProperties (    IntPtr                      hdConference,
                                           ConferencePropertyType      PropertyType,
                                           ref ConferenceSupervisor    PropertyValue )
```

Parameters

hdConference

[in] The *hdConference* parameter identifies the conference previously created by DivaCreateConference.

PropertyType

[in] The *PropertyType* parameter specifies the property to be retrieved. For more information on the available properties for setting, see the Remarks section.

PropertyValue (UInt32); PropertyValue (ConferenceMemberRights); PropertyValue (ConferenceSupervisor)

[in] This parameter contains the new value for the property type.

Return values

If the function succeeds, the return value is *DSAPI.ReturnCode.Success* (0). Other possible return values are *DSAPI.ReturnCode.ErrorInvalidParameter* and *DSAPI.ReturnCode.InvalidHandle*.

Remarks

The function writes the specified conference property. The available conference properties and the corresponding data types are:

DSAPI.ConferencePropertyType.	Data Type
MaxMembers	UInt32
Options	UInt32 (DSAPI.ConferenceOptions)
DefRights	UInt32 (DSAPI.ConferenceRights)
MemberRights	DSAPI.ConferenceMemberRights
Supervisor	DSAPI.ConferenceSupervisor
NumMembers	N/A for set function
Members	N/A for set function
NumTalkers	N/A for set function
Talkers	N/A for set function

CHAPTER 5

SMS support in the Dialogic® Diva® API .NET wrapper

The Dialogic® Diva® API supports sending and receiving of SMS short messages. The C header file smssdk.h contains the declarations of the SMS module. The header files also contain the documentation of the functions and defines.

The .NET wrapper contains an adaptation of this interface to the .NET platform. The following describes the differences between the Diva API and the Diva API for .NET. This documentation should be used together with the documentation in the Dialogic® Diva® API header file.

Adaptation of constants, data types, and functions

The structures and functions are contained inside a static class named `SMS1` which is contained inside the `DSAPI` class of the .NET wrapper. The following examples show the naming scheme.

smssdk.h name	Diva API for .NET
SMS1_MAX_NUMBER_OF_EVENTS	DSAPI.SMS1.MAX_NUMBER_OF_EVENTS
sms_error_t	DSAPI.SMS1.Error
SMS1_ERROR_UT_INIT_FAIL	DSAPI.SMS1.Error.UT_INIT_FAIL
sms1_tx_deliver_pdu_t	DSAPI.SMS1.TXDeliverPDU
DivaSms1Start()	DSAPI.SMS1.Start()
DivaSms1NewCall()	DSAPI.SMS1.NewCall()

The names of structures were changed to camel notation; the members retained their original names, common prefixes are removed.

Sending and receiving of messages and events

The polymorphic data structure `sms1_app_message_t`, which contains a union, is not directly representable using .NET data types. Therefore a different solution is provided by the .NET wrapper.

Sending of messages

The Dialogic® Diva® API function *DivaSms1SendMsgToL3()* is used to send messages to the SMS module of the Diva API. These messages are used to establish/release a logical link, and to send data units (PDUs) over the telephone line.

In the Dialogic® Diva® API for .NET wrapper, this function has been replaced by a set of methods:

Message type	Diva API for .NET function
SMS1_TL_EST_IND	DSAPI.SMS1.LinkEstablish()
SMS1_TL_SMS_REL	DSAPI.SMS1.LinkRelease()
SMS1_TL_SMS_IND	DSAPI.SMS1.SendSubmitPDU() DSAPI.SMS1.SendDeliverPDU() DSAPI.SMS1.SendCommandPDU() DSAPI.SMS1.SendStatRepPDU() DSAPI.SMS1.SendSubmitReportPDU() DSAPI.SMS1.SendDeliverReportPDU()

Receiving of events

The Dialogic® Diva® API event *DivaEventSms1MsgReceived* notifies the application of certain events, such as a reject cause, or the reception of a PDU. The event carries a pointer to a `sms1_app_message_t` structure, which in the .NET environment is expressed as an `IntPtr`. In order to relieve the application of the task of marshalling this pointer to a .NET data structure a different mechanism has been devised.

The function *DSAPI.SMS1.TranslateAndDispatchMessage()* receives the `IntPtr` of the message structure and an application-implemented interface whose methods are then called depending on the message contents.

The member functions of the *DSAPI.SMS1.MessageHandler* interface correspond to the event types as follows:

Event type	MessageHandler interface method
SMS1_APP_SMS_IND	OnCommand() OnSubmit() OnDeliver() OnSubmitReport() OnDeliverReport() OnStatusReport()
SMS1_CM_REL_IND	OnReleaseInd()
SMS1_APP_SMS_REJ	OnRejected()
SMS1_APP_DL_ERROR	OnDataLinkError()

Sample code

The sample SMSServiceCenter is available for the Dialogic® Diva® API and for the Dialogic® Diva® API for .NET. The functionality of both samples is identical. The sample using the Diva API for .NET is based on the C# framework available in source. Please refer to the C# framework for more information on the SMS implementation.