Dialogic

Dialogic® MSML Media Server Software

User's Guide

March 2012

05-2513-008

Publication Date: March 2012

Document Number: 05-2513-008

Contents

1 MSML Media Server Software Overview 15

1.1 Introduction 15
1.2 Media Server Operating Model 15

2 Configuration 17

2.1 media_server Application 17
2.2 mediaserver Configuration Script 17
2.3 Configuring Schema Validation 18
2.4 media_server.xml Configuration File 18
Example of Default media_server.xml File 20

3 Feature and Protocol Package Support 23

3.1 Feature Highlights 23
3.2 Media Server Markup Language (MSML) Protocol Package Support 24
3.3 MSML Support Details 47
3.3.1 MSML Core Package Support 47
3.3.2 MSML Conference Core Package Support 47
<asn> **49**
<audiomix> **50**
<clamp> **51**
<createconference> 52
<destroyconference> **54**
<gain> **55**
<join> **57**
<modifyconference> 59
<modifystream> **61**
<n-loudest> 62
<region> **63**
<root> **67**
<selector> **68**
<stream> **69**
<unjoin> **71**
<videolayout> **72**
3.3.3 MSML Dialog Package Support 73
3.3.4 MSML Audit Package Support 73

4 Deviations from IETF RFC 75

5 Feature Details 77

5.1 SIP Session Timer 77
5.1.1 Feature Description 77
5.1.2 Enabling SIP Session Timer 77
AS: SIP INVITE request for session timer support 78
MS: SIP response to request for session timer support 78
AS: SIP UPDATE request 79

MS: SIP response to SIP UPDATE request 79
5.1.3 Using SIP Session Timer with Dialogic® HMP Software 79
5.2 SIP re-INVITE 80
5.2.1 Feature Description 81
5.2.2 Media Stream Direction Support 81
5.3 HTTP Play and Record 82
5.3.1 Feature Description 82
5.3.2 Requirements for HTTP Support 82
5.3.3 Dialog Execution 82
5.3.4 Audio and Video Playback 82
5.3.5 Audio and Video Recording 83

6 Sample Use Case 85

6.1 Use Case Description 85
6.2 MSML Control Syntax in Use Case 87
6.2.1 Establish Connections 87
6.2.2 Play Main Prompt 87
6.2.3 Play Video Portal Prompt 88
6.2.4 Play Video Clip 90
6.2.5 Record Message 92
6.2.6 Replay Main Prompt 93
6.2.7 Terminate Connections 94

7 MSML Script Examples 95

7.1 MSML Scripts for Audio Conferencing 95
7.1.1 Creating a basic audio conference with <asn> and <n-loudest> 95
7.1.2 Modifying a basic audio conference with <asn> and <n-loudest> 95
7.1.3 Joining six parties to an audio conference 96
7.1.4 Call center coach-pupil conference 96
7.1.5 Muting an audio stream flowing into a conference 97
7.1.6 Un-muting an audio stream flowing into a conference 97
7.1.7 Un-joining streams using wildcards 98
7.1.8 Continuous digit collection on a conference participant 99
7.1.9 Destroying a conference 101
7.2 MSML Scripts for Video Conferencing 101
7.2.1 Creating a four party layout video conference 101
7.2.2 Expanding a four party layout to a six party layout video conference 102
7.2.3 Contracting a six party layout to a four party layout video conference 103
7.2.4 Layered regions in a video conference 104
7.2.5 Single party layout video conference using a <selector> element 105
7.2.6 Sequencing parties through regions in a video conference layout 105
7.2.7 Recording a segment of a video conference 107
7.2.8 Playing audio into a video conference 107

8 Diagnostics 109

8.1 Overview 109
8.2 Logging Configuration 109
8.2.1 Configuration File Format 109
8.2.2 Logging Labels 110
8.2.3 Client Categories 111
8.3 Log File Format 112

A Media Server Markup Language (MSML) Overview 115
A.1 Introduction 115
A.2 MSML Elements 115
A.2.1 <msml> 116
A.2.2 <send> 116
A.2.3 <result> 116
A.2.4 <event> 116
A.3 Stream Manipulation Elements 116
A.3.1 <join> 116
A.3.2 <modifystream> 116
A.3.3 <unjoin> 116
A.4 Conference Elements 117
A.4.1 <createconference> 117
A.4.2 <modifyconference> 117
A.4.3 <destroyconference> 117
A.5 Dialog Elements 117
A.5.1 <dialogstart > 117
A.5.2 <dialogend > 117
A.6 Receiving Events from a Client 117
A.7 Sending Events and Transaction Results to a Client 118
A.7.1 Transaction Results 118
A.7.2 Events 118
A.8 Media Server Object Model 118
A.8.1 Network Connections (conn) 118
Definition 118
Object Creation 118
A.8.2 Conference (conf) 119
Definition 119
Object Creation 120
A.8.3 Dialog (dialog) 120
Definition 120
Object Creation 120
A.8.4 Operator (oper) 121
Definition 121
Object Creation 121

Glossary 123

Figures

1 Media Server Operating Environment 16
2 Audio/Video Play/Record Scenario 86
3 Network Connection Creation. 119

Tables

1 High-Level Feature Summary 24
2 MSML Protocol Supported Packages 25
3 MSML Core Package Support 26
4 MSML Conference Core Package Support 26
5 MSML Dialog Core Package Support 30
6 MSML Dialog Base Package Support 32
7 MSML Dialog Group Package Support 39
8 MSML Dialog Transform Package Support 39
9 MSML Audit Core Package Support 41
10 MSML Audit Conference Package Support 41
11 MSML Audit Connection Package Support 43
12 MSML Audit Dialog Package Support 45
13 MSML Audit Stream Package Support 46

Revision History

This revision history summarizes the changes made in each published version of this document.

Document No.	Publication Date	Description of Revisions
05-2513-008	March 2012	Feature and Protocol Package Support chapter: In Table 4. MSML Conference Core Package Support, Section 8.12.2.1, added comments to the <clamp> element and the dtmf attribute.
05-2513-007	February 2012	Feature and Protocol Package Support chapter: Changed the <n-loudest> element from not supported to supported in Table 4. Added a section about MSML Support Details and added several element description to MSML Conference Core Package Support. Deviations from IETF RFC chapter: Removed the <unjoin> element from the fourth bullet because wildcards for the <unjoin> element are now supported. Removed the last bullet because the notice about amt no longer applies. MSML Script Examples chapter: Added this new chapter.
05-2513-006	October 2011	Removed Dialogic® Multimedia Software for AdvancedTCA from the document because this product is no longer supported. Replaced <msml version="1.0"> with <msml version="1.1"> throughout the code. About This Publication chapter: Updated Purpose and Scope sections. Configuration chapter: Added the media_server.xml Configuration File section. Feature and Protocol Package Support chapter: Updated this chapter in its entirety. Added the following tables, MSML Audit Core Package Support, MSML Audit Conference Package Support, MSML Audit Connection Package Support, MSML Audit Dialog Package Support, and MSML Audit Stream Package Support. Media Server Markup Language (MSML) Overview chapter: Updated the Introduction section.
05-2513-005	December 2010	Updated for HMP 4.1LIN support. High-Level Feature Summary table: Updated to show support for Dialogic® HMP Software 4.1LIN. Changed the PSTN row for Dialogic® HMP Software 3.0WIN and Dialogic® HMP Software 3.1LIN to “D” for deprecated. Added “D” for deprecated to the table legend. Configuration chapter: added a paragraph about what to do after changing the media_server startup mode in the mediaserver Configuration Script section. (IPY00080612) Client Categories section: Added OFFER_ANSWER to the SIP Call Control category and added PORT_STREAM to the Connections category.
05-2513-004	September 2008	High-Level Feature Summary table: Updated to show PSTN not supported for Dialogic® HMP Software 3.0WIN and Dialogic® HMP Software 3.1LIN.
05-2513-003	February 2008	High-Level Feature Summary table: Updated to reflect PSTN support for Dialogic® HMP Software 3.0WIN and Dialogic® HMP Software 3.1LIN and Video support for Dialogic® HMP Software 3.0WIN.

Document No.	Publication Date	Description of Revisions
05-2513-002	December 2007	About This Publication chapter: Updated Scope section (audio conferencing is supported). Configuration chapter: Updated media_server Application section and mediaserver Configuration Script section to identify Linux specific information and add Windows information. Updated Configuring Schema Validation section to include msml.xsd and information on enabling/disabling feature on Windows® systems. Sample Use Case chapter: Added I-frame information to Step 5G in Record Message section. Updated Audio/Video Play/Record Scenario figure to include step 5G. Feature and Protocol Package Support chapter: Added Feature Highlights section. Renamed title of MSML and MOML sections from "...Feature Support Matrix" to "...Protocol Support Matrix". Feature Details chapter: New chapter. Includes SIP Session Timer, SIP re-INVITE, and HTTP Play and Record (the last section also includes HTTP requirements information which was moved from Feature Support chapter).
05-2513-001	October 2007	Made global changes to reflect Dialogic brand. Changed title from "MSML/MOML Media Server Software User's Guide" to "MSML Media Server Software User's Guide" and removed certain references to MOML within the document. Configuration chapter: Added new section: Configuring Schema Validation. MOML Feature Support Matrix table: Changed the postspeech attribute in the <record> section to supported.
05-2513-001-03	August 2006	General: Updates for "http://" scheme support. MSML Feature Support Matrix table: Updated to reflect conference support. Deviations from IETF Draft chapter: Added two items related to conference support. Configuration File Format section: Added MEDIA_CODER component for RTF logging. Client Categories section: Updates to reflect conference support. Conference Elements section: Added brief descriptions of the conference elements. Media Server Object Model section: Added a subsection describing the "Conference" object.
05-2513-001-02	June 2006	Updates for diagnostics integration with RTF.
05-2513-001-01	April 2006	Initial version of document.

About This Publication

The following topics provide information about this publication.

- Purpose
- Scope
- Intended Audience
- How to Use This Publication
- Related Information

Purpose

This publication documents the Media Server Markup Language (MSML) Media Control Interface software that provides an interface between an application server (AS) and Dialogic's host-based PowerMedia Server.

This publication is for users of the MSML Media Server Software who choose to write applications that require remote control management of MS resources available on platforms running Dialogic® PowerMedia™ Host Media Processing (HMP) software.

Additionally, this publication documents Dialogic's compliance with the IETF RFC 5705 MSML specification, describing extensions, deviations, and/or omissions from the standard. RFC 5707 is considered the normative implementation reference that readers and developers should consult in conjunction with this guide.

Scope

The MSML Media Server Software functionality is being introduced in a phased approach. A phase typically introduces support for a previously unsupported package(s), element(s) or attribute(s). This manual documents the functionality provided by the current set of supported MSML packages as described in RFC5707 as implemented in this version of the MSML Media Server Software. This includes the following packages:

- MSML Core Package
- MSML Conferencing Core Package
- MSML Dialog Core Package
- MSML Dialog Base Package
- MSML Dialog Group Module Package (parallel topology only)
- MSML Dialog Transform Primitives ModulePackage (gain only)

Functionality that is not supported by the current implementation phase includes:

- MSML Dialog Speech Package
- MSML Dialog Fax Detect Package
- MSML Dialog Fax Send / Receive Package

Future implementation phases are planned to provide additional MSML support.

Intended Audience

This publication is for:

- System Integrators
- Independent Software Vendors (ISVs)
- Value Added Resellers (VARs)
- Original Equipment Manufacturers (OEMs)

This publication assumes that the reader is familiar with the Session Initiation Protocol (SIP) as defined in IETF RFC3261.

How to Use This Publication

This manual is divided into the following sections:

- Chapter 1, "MSML Media Server Software Overview" describes the role of the MSML Media Server Software in a Media Server environment.
- Chapter 2, "Configuration" explains how to configure the MSML Media Server Software for operation on a Media Server.
- Chapter 3, "Feature and Protocol Package Support" specifies high-level feature support by platform and identifies packages, elements, and attributes (as documented in the MSML IETF standard RFC5707) currently supported or not supported by the MSML Media Server Software.
- Chapter 4, "Deviations from IETF RFC" explains deviations from the MSML IETF RFC 5707 specification.
- Chapter 5, "Feature Details" provides details on features supported, including a feature description and how-to information.
- Chapter 6, "Sample Use Case" presents an application that demonstrates many of the features currently supported by the MSML Media Server Software.
- Chapter 8, "Diagnostics" describes the logging capabilities available to the MSML Media Server Software for diagnostic purposes.
- Appendix A, "Media Server Markup Language (MSML) Overview" provides a high-level introduction to MSML.
- A **Glossary** can be found at the end of the document.

Related Information

See the following for additional information:

- *http://www.dialogic.com/manuals/* (for Dialogic® product documentation)
- *http://www.dialogic.com/support/* (for Dialogic technical support)
- *http://www.dialogic.com/* (for Dialogic® product information)

MSML Media Server Software Overview 1

This chapter provides an overview of the MSML Media Server Software. Topics include:

- Introduction 15
- Media Server Operating Model 15

1.1 Introduction

The MSML Media Server Software is an integral part of the system software provided by Dialogic® Host Media Processing (HMP) Software Release 4.1LIN.

When the Dialogic® system software is installed on a media server (MS), the MSML Media Server Software enables a remote client, also known as an application server (AS), to control media resources.

Note: The MSML Media Server Software is based on the evolving Media Server Markup Language (MSML) as defined in the MSML IETF RFC 5707, which combines the original MSML and Media Object Markup Language (MOML) drafts.

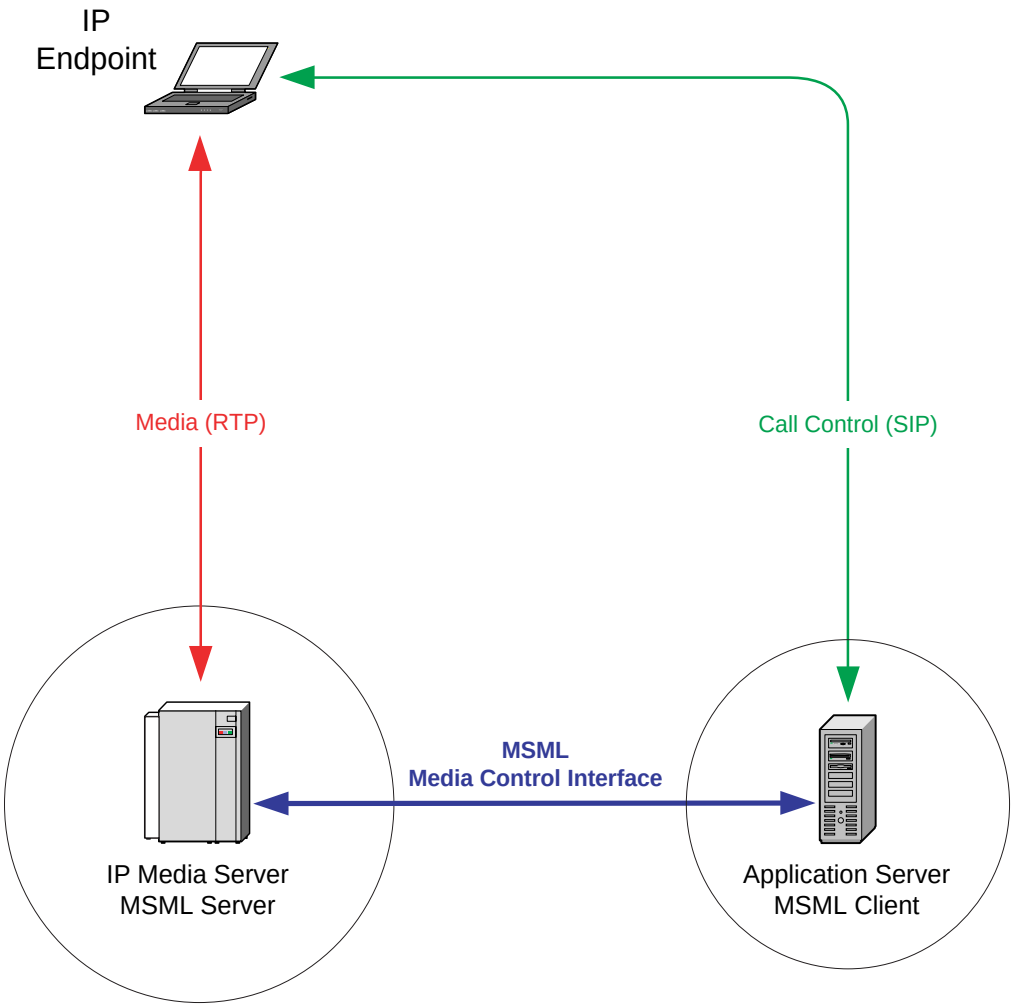
The connection between the AS and MS is established using the SIP protocol; thereafter, media control commands/responses (in the form of MSML control syntax) are exchanged in SIP messages, such as the INFO message or the 200 OK response.

1.2 Media Server Operating Model

Figure 1 shows an environment where the media server (MS) and application server (AS) operate as separate entities. The MSML Media Server Software runs on the MS and provides the interface between the AS and the MS as shown. The MS is responsible for media processing only; call control is the responsibility of the AS.

The AS, as a MSML client, must be capable of interpreting and generating MSML control syntax and must support the SIP INVITE, 200 OK, ACK, BYE and INFO messages.

Figure 1. Media Server Operating Environment



Configuration 2

This chapter discusses configuration topics, such as how to configure a media server to use the MSML Media Server Software.

- media_server Application 17
- mediaserver Configuration Script 17
- Configuring Schema Validation 18
- media_server.xml Configuration File 18

2.1 media_server Application

The *media_server* application runs on the media server and provides the MSML Media Server Software functionality.

On Linux, the *media_server* application can be configured using the *mediaserver* configuration script as described in Section 2.2, “mediaserver Configuration Script”, on page 17.

On Windows, the *media_server* application runs as a Windows service.

2.2 mediaserver Configuration Script

On Linux, the *mediaserver* configuration script is used to enable/disable and start/stop the *media_server* application and is located in the *bin* directory.

The *mediaserver* configuration script provides the following options:

mediaserver enable
: Enables the *media_server* application. All subsequent calls to *dlstart* start the *media_server* application. All subsequent calls to *dlstop* stop the *media_server* application.

mediaserver disable
: Disables the *media_server* application. Subsequent calls to *dlstart* do **not** start the *media_server* application.

mediaserver start
: Starts the *media_server* application. The *media_server* application must be started after *dlstart* successfully completes.

mediaserver stop
: Stops the *media_server* application.

After changing the media_server startup mode, and before starting the Dialogic® system (via `dlstart`), logout and then login, or open a new shell and issue `dlstart`.

2.3 Configuring Schema Validation

When enabled, the schema validation functionality included in the product validates each and every XML body received from the application server against a pre-installed MSML schema, *msml.xsd*, located in the *cfg* directory. The inbound XML body must pass the schema validation before it will be executed by the media server. This is especially useful for application developers during the development process. It ensures that the syntax and attribute definitions in the XML body are correct and match the supported functionality in the schema.

- To reduce CPU utilization, this functionality is disabled by default. It is recommended that it remain disabled in a production runtime environment.
- To enable this functionality, edit the *media_server.xml* configuration file, which is located in the *cfg* directory.

2.4 media_server.xml Configuration File

Additional customization of the MSML media server is available by modifying the *media_server.xml* configuration file. This file is located in the cfg directory and is used by the media server during initialization to configure feature and system parameters. The following configurable parameters are supported:

- ConferenceAGC
  - Controls automatic gain control (AGC) of the audio stream into the conference.
  - Values: ENABLE, DISABLE
  - Default: ENABLE
- MEDIA_CONTROL_PROTOCOL_APP_SRV_ACCESS
  - Controls access to the MSML media server. When enabled, only trusted application servers that are listed in the “TrustedAppList” table can access the media server.
  - Values: ENABLE, DISABLE
  - Default: DISABLE (this allows access for all application hosts)
- TrustedAppList
  - Contains the IP addresses of all trusted servers that have authorization to access the MSML media server.
  - Values: IP address name and dot-notation IP address pair for each entry
  - Default: Empty
- MEDIA_CONTROL_PROTOCOL_SIP_IPADDR
  - IP Address of the MSML media server
  - Values: IP address name and dot-notation IP address pair
  - Default: Set by the MSML media server
- MEDIA_CONTROL_PROTOCOL_SIP_PORT
  - SIP port used for control messages

  - Values: valid port #
  - Default: 5060
- MEDIA_CONTROL_PROTOCOL_SIP_TRANSPORT
  - SIP Transport protocol
  - Values: UDP, UDP_TCP
  - Default: UDP
- HTTPCaching
  - Controls a caching mechanism to improve performance when servicing network/remote file operations.
  - Values: ENABLE, DISABLE
  - Default: DISABLED (Do not perform caching; all network request will result in remote access.)
- SchemaValidation
  - Controls activation of the XML validation of the media control message body. Validation is performed based on the msml.xsd XML schema definition file. This feature is MIPs intensive and is recommended during application development and trouble shooting. Not recommended for normal operation.
  - Values: ENABLE, DISABLE
  - Default: DISABLE
- DefaultStorageDirectory
  - Defines the local directory used to store recorded files. This is the default directory when an explicit directory is not defined in the msml request.
  - Values: local directory
  - Default: C:\MediaServerRecordings
- MSMLVersion
  - MSMLversion used by the media server
  - Values: 1.0, 1.1
  - Default: 1.1
- ContentType
  - MSML control package Content Type
  - Values: xml, msml-xml
  - Default: xml
- Encoding
  - XML encoding
  - Values: utf_8, us_ascii
  - Default: utf_8
- VideoFastUpdate
  - Specifies the control method for receiving Video Fast Update request.
  - Values: INFO, DISABLE
  - Default: INFO (request sent in SIP INFO message)
- ResourceUsage
  - Controls the media devices that are reserved for use by the media server. The media server will control the first 1 to N resources of a specified resources. N must be less than or

equal to the maximum defined for each resource type contained in the HMP license. For example, if 100 IPM resources are defined in the license and 50 IPM resources are specified in this section, the MSML media server will control the first 50 resources (i.e. 1-50). The remaining 51-100 IPM resources are available for use by another application.

– Values: -1, 1-N where N is less than or equal to maximum as defined by the HMP license.

– Default: -1 (all resources reserved for use by the MSML media server)

Example of Default media_server.xml File

```
<?xml version="1.0"?>
<MEDIA_CONTROL_CONFIG>
<Table>VirtualBoardConfig
<row>Board_1
<Param>
<!-- ENABLE: Only allow access to application hosts that are in the Trusted App List  -->
<!-- DISABLE: allows access for all application hosts -->
<name>MEDIA_CONTROL_PROTOCOL_APP_SRV_ACCESS</name>
<ParamValue>DISABLE</ParamValue>
</Param>
<Param>
<!-- DEFAULT: SIP IP Address will be determined by the server -->
<!-- SIP IP Address format has to be like: 192.168.1.1 -->
<name>MEDIA_CONTROL_PROTOCOL_SIP_IPADDR</name>
<ParamValue>DEFAULT</ParamValue>
</Param>
<Param>
<!-- SIP PORT -->
<name>MEDIA_CONTROL_PROTOCOL_SIP_PORT</name>
<ParamValue>5060</ParamValue>
</Param>
<Param>
<!-- SIP TRANSPORT either UDP or UDP_TCP -->
<name>MEDIA_CONTROL_PROTOCOL_SIP_TRANSPORT</name>
<ParamValue>UDP</ParamValue>
</Param>
</row>
</Table>
<Table>TrustedAppList
<!-- Commented example of the syntax required to list a Trusted App IP address.
<row>
<Param>
<name>IPAddress</name>
<ParamValue>146.152.1.1</ParamValue>
</Param>
</row>
-->
</Table>
<AdaptorPort>32868</AdaptorPort> For Internal Use
<!-- HTTP Caching - either ENABLE or DISABLE -->
<HTTPCaching>DISABLE</HTTPCaching>
<!-- Schema Validation - either ENABLE or DISABLE -->
<SchemaValidation>DISABLE</SchemaValidation>
<!-- Default Storage Directory - Fully qualified directory path used as a default directory for
storage -->
<!-- Commented examples of using the Default Storage Directory. -->
<DefaultStorageDirectory>C:\MediaServerRecordings</DefaultStorageDirectory>
<!-- MSMLVersion - either 1.0 or 1.1 -->
<MSMLVersion >1.1</MSMLVersion>
<!-- ContentType - either xml or msml_xml -->
<ContentType >xml</ContentType>
<!-- Encoding - either utf_8 or us_ascii -->
<Encoding >utf_8</Encoding>
<!-- ClearDB - either ENABLE or DISABLE -->For Internal Use Only
```

```
<ClearDB>DISABLE</ClearDB>  For Internal Only
<!-- DTMFStartTimer - either ENABLE or DISABLE --><DTMFStartTimer>DISABLE</DTMFStartTimer> For
 Internal Only

<!-- AdvDigitPattern - either ENABLE or DISABLE -->
<AdvDigitPattern>DISABLE</AdvDigitPattern> For Internal Only
<!-- VideoFastUpdate - either INFO or DISABLE -->
<VideoFastUpdate >INFO</VideoFastUpdate >
<!-- Explicitly define the maximum number of each resource type
     that is available to the media server.
     -1 = use all resources of that type. -->
<ResourceUsage>
   <MaxIpm>-1</MaxIpm>
     <MaxIpt>-1</MaxIpt>
   <MaxDx>-1</MaxDx>
   <MaxMm>-1</MaxMm>
</ResourceUsage>
</MEDIA_CONTROL_CONFIG>
```

3 Feature and Protocol Package Support

This chapter describes the high-level features supported by the current version of the MSML Media Server Software, MSML protocol support, and other related topics.

- Feature Highlights 23
- Media Server Markup Language (MSML) Protocol Package Support 24
- MSML Support Details 47

3.1 Feature Highlights

The MSML Media Server Software level of support varies in conjunction with the associated Dialogic Host Media Processing platform which it uses to provide media operations and services. Table 1 presents the high-level features and functionality supported in the current version of the MSML Media Server Software with respect to Dialogic platforms.

As new features and functionality are introduced, this table will be updated to reflect the latest supported capability.

Table 1. High-Level Feature Summary

Feature	Dialogic® HMP Software 4.1LIN	Feature Details / Comments
Audio conferencing	S	
Audio play and record	S	
Audit package	S	
Digit detection - inband	S	
Digit detection - RFC 2833	S	
HTTP play and record	S	See Section 5.3, "HTTP Play and Record", on page 82.
I-frame	S	See Section 6.2.5, "Record Message", on page 92, step 5G. I-frame timeout notification supported; I-frame on-demand not yet supported.
SIP re-INVITE	S	See Section 5.2, "SIP re-INVITE", on page 80
SIP session timers (keepalive)	S	See Section 5.1, "SIP Session Timer", on page 77.
Video play and record	S	
Video conferencing	S	
Legend: S = supported; NS = not supported Dialogic® HMP Software 4.1LIN = Dialogic® Host Media Processing Software Release 4.1LIN		

3.2 Media Server Markup Language (MSML) Protocol Package Support

The following section describes the current level of support for MSML Packages and the elements and attributes defined within each MSML Package. Supported items are shown in black text; unsupported items are shown in red text. The "Comment" column indicates restrictions or limitations that the current version of the MSML Media Server Software imposes.

Table 2 shows the high-level support view for the complete set of packages as defined in RFC5707. Additional details are provided in subsequent tables describing element and attribute level support for each package available.

Note: The level of support is correlated against the IETF standard RFC 5707 Media Server Markup Language.

Table 2. MSML Protocol Supported Packages

RFC 5707 Ref	Package Name	Requirement	Level of Support
Section 7	MSML Core Package	Mandatory	Supported, see Table 3, "MSML Core Package Support", on page 26.
Section 8	MSML Conference Core Package	Conditionally Mandatory, for Conferencing	Supported, see Table 4, "MSML Conference Core Package Support", on page 26.
Section 9.6	MSML Dialog Core Package	Conditionally Mandatory, for Dialogs	Supported, see Table 5, "MSML Dialog Core Package Support", on page 30.
Section 9.7	MSML Dialog Base Package	Conditionally Mandatory, for Dialogs	Supported, see Table 6, "MSML Dialog Base Package Support", on page 32.
Section 9.8	MSML Dialog Group Package	Optional	Supported, see Table 7, "MSML Dialog Group Package Support", on page 39.
Section 9.9	MSML Dialog Transform Package	Optional	Supported, see Table 8, "MSML Dialog Transform Package Support", on page 39.
Section 9.10	MSML Dialog Speech Package	Optional	Not Supported
Section 9.11	MSML Dialog Fax Detection Package	Optional	Not Supported
Section 9.12	MSML Dialog Fax Send/Receive Package	Optional	Not Supported
Section 10.1	MSML Dialog Audit Core Package	Conditionally Mandatory, for Auditing	Supported, see Table 9, "MSML Audit Core Package Support", on page 41.
Section 10.2	MSML Audit Conference Package	Conditionally Mandatory, for Auditing Conference, Conference Dialog, and Conference Stream	Supported, see Table 10, "MSML Audit Conference Package Support", on page 41.
Section 10.3	MSML Audit Connection Package	Conditionally Mandatory, for Auditing Connection, Connection Dialog, and Connection Stream	Supported, see Table 11, "MSML Audit Connection Package Support", on page 43.
Section 10.4	MSML Audit Dialog Package	Conditionally Mandatory, for Auditing Dialog	Supported, see Table 12, "MSML Audit Dialog Package Support", on page 45.
Section 10.5	MSML Audit Stream Package	Conditionally Mandatory, for Auditing Stream	Supported, see Table 13, "MSML Audit Stream Package Support", on page 46.

Table 3. MSML Core Package Support

RFC 5707 Ref	Element Name	Attribute Name	Level of Support	Comments
Section 7.1	<msml>	-	supported	
		version	supported	Can be “1.0” or “1.1”, mutually exclusive.
Section 7.2	<send>	-	supported	
		event	supported	
		target	supported	
		valuelist	supported	
		mark	supported	
Section 7.3	<result>	-	supported	
		response	supported	
		mark	supported	
Section 7.4	<event>	-	supported	
		name	supported	
		id	supported	

Table 4. MSML Conference Core Package Support

RFC 5707 Ref	Element Name	Attribute Name	Level of Support	Comments
Section 8.3	<createconference>	-	supported	
		name	supported	
		deletewhen	supported	
		term	supported	
		mark	supported	
Section 8.3.1	<reserve>	-	not supported	
		required	not supported	
Section 8.3.1.1	<resource>	-	not supported	
		n	not supported	
		type	not supported	
Section 8.4	<modifyconference>	-	supported	
		id	supported	
		mark	supported	
Section 8.5	<destroyconference>	-	supported	
		id	supported	
		mark	supported	

Table 4. MSML Conference Core Package Support

RFC 5707 Ref	Element Name	Attribute Name	Level of Support	Comments
Section 8.6	<audiomix>	-	supported	
		id	supported	
		samplerate	not supported	
Section 8.6.1	<n-loudest>	-	supported	
		n	supported	
Section 8.6.2	<asn>	-	supported	
		ri	supported	
		asth	not supported	
Section 8.7	<videolayout>	-	supported	
		type	supported	
		id	supported	
Section 8.7.1	<root>	-	supported	
		size	supported	
		backgroundcolor	not supported	
		backgroundimage	not supported	

Table 4. MSML Conference Core Package Support

RFC 5707 Ref	Element Name	Attribute Name	Level of Support	Comments
Section 8.7.2	<region>	-	supported	
		id	supported	
		left	supported	
		top	supported	
		relativesize	supported	
		priority	supported	
		title	not supported	
		titletextcolor	not supported	
		titlebackgroundcolor	not supported	
		bordercolor	not supported	
		borderwidth	not supported	
		logo	not supported	
		freeze	not supported	
		blank	not supported	
Section 8.7.4	<selector>	-	supported	Supported only for single party layout, no regions defined.
		id	supported	
		method	supported	Supports “vas” only.
		status	not supported	
		blankothers	not supported	

Table 4. MSML Conference Core Package Support

RFC 5707 Ref	Element Name	Attribute Name	Level of Support	Comments
Section 8.7.3.1	<vas>	-	not supported	
		si	not supported	
		speakersees	not supported	
Section 8.8	<join>	-	supported	
		id1	supported	
		id2	supported	
		mark	supported	
Section 8.9	<modifystream>	-	supported	
		id1	supported	
		id2	supported	
		mark	supported	
Section 8.19	<unjoin>	-	supported	
		id1	supported	
		id2	supported	
		mark	supported	
Section 8.11	<monitor>	-	not supported	
		id1	not supported	
		id2	not supported	
		compressed	not supported	
Section 8.12	<stream>	-	supported	
		media	supported	
		dir	supported	
		compressed	not supported	
Section 8.12.1 Audio Stream Properties		preferred	supported	
Section 8.12.1.1 Video Stream Properties		display	supported	
		override	not supported	

Table 4. MSML Conference Core Package Support

RFC 5707 Ref	Element Name	Attribute Name	Level of Support	Comments
Section 8.12.1.1	<gain>	-	supported	
		id	not supported	
		amt	supported	
		agc	supported	
		tgtlvl	not supported	
		maxgain	not supported	
Section 8.12.2.1	<clamp>	-	supported	Limited to audio where the stream direction is to a conference ID.
		dtmf	supported	Mandatory field; can be set to “true” or “false”.
		tone	supported	Mandatory field; must always be set to “false”.
Section 8.12.2.2	<visual>	-	not supported	

Table 5. MSML Dialog Core Package Support

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
Section 9.6.1	<dialogstart>	-	supported	
		attributes:	-	
		target	supported	
		src	supported	Supports the following schemes: • “file://” • “http://”
		type	supported	Vxml not supported; only moml
		name	supported	
		mark	supported	
Section 9.6.2	<dialogend>	-	supported	
		attributes:		
		id	supported	
		mark	supported	

Table 5. MSML Dialog Core Package Support

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
Section 9.6.3	<send>		supported	
		attributes:		
		event	supported	
		target	supported	
		namelist	supported	
Section 9.6.4	<exit>		not supported	
		attributes:		
		namelist	not supported	
Section 9.6.5	<disconnect>		not supported	
		attributes:		
		namelist	not supported	

Table 6. MSML Dialog Base Package Support

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
Section 9.7.1	<play>	-	supported	
		attributes:	-	
		id	supported	
		interval	supported	
		iterate	supported	
		initial	supported	
		maxtime	supported	
		barge	supported	No effect for video.
		cleardb	supported	Supports “true” only. No effect for video.
		offset	supported	No effect for video.
		skip	not supported	
		xml:lang	not supported	
		events:	-	
		pause	not supported	
		resume	supported	
		forward	not supported	
		backward	not supported	
		restart	supported	
		toggle-state	not supported	
		terminate	supported	
		shadow variables:	-	
		play.amt	supported	No support for video.
		play.end	supported	

Table 6. MSML Dialog Base Package Support

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
Section 9.7.1.1	<audio>	-	supported	
		attributes:		
		uri	supported	Supports the following schemes: “file://” “http://”
		format	supported	Supports “audio/wav” and “audio/vox;codecs=value” only. Codecs are ignored for wav format. Codecs are required for vox format; valid codecs are: “mulaw”, “alaw”, “pcm”, “dialogic_adpcm”, “g726”
		audiosamplerate	supported	Ignored for wav format. Required for vox format; valid values are: 6, 8, 11
		audiosamplesize	supported	Ignored for wav format. Required for vox format; valid values are: 2, 4, 8
		iterate	supported	
		xml:lang	not supported	

Table 6. MSML Dialog Base Package Support

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
Section 9.7.1.2	<video>	-	supported	
		attributes:		
		uri	supported	Supports the following schemes: “file://” “http://”
		format	supported	Supports “video/vid;codecs=h263” and “video/vid;codecs=h264.
		audiosamplerate	not supported	
		audiosamplesize	not supported	
		codecconfig	not supported	
		profile	supported	No values defined.
		level	supported	No values defined.
		imagewidth	supported	No values defined.
		imageheight	supported	No values defined.
		maxbitrate	supported	No values defined.
		framerate	supported	No values defined.
		iterate	supported	
Section 9.7.1.3	<media>	-	supported	
Section 9.7.1.4	<var>	-	not supported	
		attributes:		
		type	not supported	
		subtype	not supported	
		value	not supported	
		xml:lang	not supported	
Section 9.7.1.5	<playexit>	-	supported	

Table 6. MSML Dialog Base Package Support

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
Section 9.7.2	<dtmfgen>	-	supported	
		attributes:		
		id	supported	
		digits	supported	
		level	supported	
		dur	supported	
		interval	supported	
		shadow variables:		
		dtmfgen.end	supported	
Section 9.7.2.1	<dtmfgenexit>	-	supported	
Section 9.7.3	<tonegen>	-	not supported	
		attributes:	-	
		id	not supported	
		iterate	not supported	
		events:	-	
		terminate	not supported	
		shadow variables:	-	
		tonegen.end	not supported	
Section 9.7.3.1	<tone>	-	not supported	
		attributes:	-	
		duration	not supported	
		iterate	not supported	
	<tone1>	-	not supported	
		attributes:	-	
		freq	not supported	
		atten	not supported	
	<tone2>	-	not supported	
		attributes:	-	
		freq	not supported	
		atten	not supported	
Section 9.7.3.2	<silence>	-	not supported	
		attributes:	-	
		duration	not supported	
Section 9.7.3.3	<tonegenexit>	-	not supported	

Table 6. MSML Dialog Base Package Support

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
Section 9.7.4	<record>	-	supported	
		attributes:	-	
		id	supported	
		append	not supported	
		dest	supported	Uses dx_ device. Supports the following schemes: “file://” “http://”
		audiodest	supported	Uses mm_ device. Supports the following schemes: “file://” “http://”
		videodest	supported	Uses mm_ device. Supports the following schemes: “file://” “http://”
		format	supported	
		codeconfig	not supported	
		audiosamplerate	supported	Valid values are: 6, 8, 11.
		audiosamplesize	supported	Valid values are: 2, 4, 8.
		profile	supported	No values defined.
		level	supported	No values defined.
		imagewidth	supported	No values defined.
		imageheight	supported	No values defined.
		maxbitrate	supported	No values defined.
		framerate	supported	No values defined.
		initial	supported	
		maxtime	supported	
		prespeech	not supported	
		postspeech	supported	
		termkey	supported	No effect for video.

Table 6. MSML Dialog Base Package Support

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
Section 9.7.4	<record>	**events:**	-	
		pause	not supported	
		resume	supported	
		toggle-state	not supported	
		terminate	supported	
		terminate.cancelled	not supported	
		terminate.finalsilence	not supported	
		nospeech	not supported	
		shadow variables:	-	
		record.len	not supported	
		record.end	not supported	
		record.recordid	not supported	
	<play> (child)	-	not supported	See <play> definition.
	<tonegen> (child)	-	not supported	See <tonegen> definition.
Section 9.7.4.3	<recordexit>	-	supported	

Table 6. MSML Dialog Base Package Support

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
Section 9.7.5	<dtmf>, <collect>	-	supported	
		attributes:	-	
		id	supported	
		cleardb	supported	Supports “true” only.
		fdt	supported	
		idt	supported	
		edt	not supported	
		starttimer	supported	
		iterate	supported	
		idd	not supported	
		events:	-	
		starttimer	supported	
		terminate	supported	
		shadow variables:	-	
		dtmf.digits	supported	
		dtmf.len	supported	
		dtmf.last	supported	
		dtmf.end	supported	
	<play> (child)	-	not supported	See <play> definition.
Section 9.7.5.2	<pattern>	-	supported	
		attributes:	-	
		digits	supported	
		format	supported	Supports the “moml+digits” format only; the “mgcp” and “megaco” formats are not supported.
		iterate	supported	
Section 9.7.5.3	<detect>	-	supported	
Section 9.7.5.4	<noinput>	-	supported	
		attributes:	-	
		iterate	supported	

Table 6. MSML Dialog Base Package Support

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
Section 9.7.5.5	<nomatch>	-	supported	
		attributes:	-	
		iterate	supported	
Section 9.7.5.6	<dtmfexit>	-	supported	
Section 9.7.6	<moml>	-	supported	
		attributes:	-	
		version	supported	Must be 1.0.
		id	supported	
		events:	-	
		terminate	supported	

Table 7. MSML Dialog Group Package Support

RFC 5707 Ref	Element Name	Attribute/Event Name	Level of Support	Comments
Section 9.8.1	<group>	-	supported	
		attributes:	-	
		topology	supported	Supports “parallel” topology only.
		id	supported	
		events:	-	
		terminate	supported	
Section 9.8.2	<groupexit>	-	supported	

Table 8. MSML Dialog Transform Package Support

RFC 5707 Ref	Element Name	Attribute/Event Name	Level of Support	Comments
Section 9.9.1	<vad>	-	not supported	
		attributes:	-	
		id	not supported	
		starttimer	not supported	
		events:	-	
		starttimer	not supported	
		terminate	not supported	

Table 8. MSML Dialog Transform Package Support

RFC 5707 Ref	Element Name	Attribute/Event Name	Level of Support	Comments
Section 9.9.1.1	<voice>, <silence>, <tvoice>, <tsilence>	-	not supported	
		attributes:	-	
		len	not supported	
		sen	not supported	
Section 9.9.2	<gain>	-	supported	
		attributes:	-	
		id	not supported	
		incr	supported	
		amt	supported	Valid values are in the range: -10 to +10.
		events:	-	
		mute	not supported	
		unmute	not supported	
		reset	supported	
		louder	supported	
		softer	supported	
		amt	supported	
Section 9.9.3	<agc>	-	not supported	
		attributes:	-	
		id	not supported	
		tgtlvl	not supported	
		maxgain	not supported	
		events:	-	
		mute	not supported	
		unmute	not supported	
Section 9.9.4	<gate>	-	not supported	
		attributes:	-	
		initial	not supported	
		events:	-	
		mute	not supported	
		unmute	not supported	
Section 9.9.5	<clamp>		not supported	
		attributes:	-	
		id	not supported	

Table 8. MSML Dialog Transform Package Support

RFC 5707 Ref	Element Name	Attribute/Event Name	Level of Support	Comments
Section 9.9.6	<relay>		not supported	
		attributes:	-	
		id	not supported	

Table 9. MSML Audit Core Package Support

RFC 5707 Ref	Element Name	Attribute Name	Level of Support	Comments
Section 10.1.1	<audit>	-	supported	
		queryid	supported	
		statelist	supported	
		mark	not supported	
Section 10.1.2	<auditresult>	-	supported	
		targetid	supported	

Table 10. MSML Audit Conference Package Support

RFC 5707 Ref	Element Name	State Parameters	Attribute Name	Level of Support	Comments
Section 10.2.1	<audit>	Prefix: “audit.conf”	-	supported	Based on Table 9 Audit Core framework.
		confconfig	-	supported	
		confconfig.audiomix	-	supported	
		confconfig.audiomix.asn	-	supported	
		confconfig.audiomix.n-loudest		not supported	
		confconfig.videolayout		not supported	
		confconfig.videolayout.root		not supported	
		confconfig.videolayout.selector		not supported	
		confconfig.controller	-	supported	
		dialog	-	supported	
		stream	-	supported	
		Child Elements			

RFC 5707 Ref	Element Name	State Parameters	Attribute Name	Level of Support	Comments
Section 10.2.2	<auditresult>	-	-	supported	Based on Table 9 Audit Core framework.
Section 10.2.2.1		confconfig	-	supported	
			deletewhen	supported	
			term	not supported	
Section 10.2.2.2		confconfig.audiomix	-	supported	
			Id	supported	
			samplerate	not supported	
Section 10.2.2.3		confconfig.audiomix.asn	-	supported	
			ri	supported	
			asth	not supported	
Section 10.2.2.4		confconfig.audiomix.n-loudest	n	not supported	
Section 10.2.2.5		confconfig.videolayout	id	not supported	
			type	not supported	
Section 10.2.2.6		confconfig.videolayout.root	size	not supported	
			backgroundcolor	not supported	
			backgroundimage	not supported	
Section 10.2.2.7		confconfig.videolayout.selector	id	not supported	
			method	not supported	
			status	not supported	
			blankothers	not supported	
			si	not supported	
			speakersees	not supported	
Section 10.2.2.8		confconfig.controller	-	supported	

RFC 5707 Ref	Element Name	State Parameters	Attribute Name	Level of Support	Comments
Section 10.2.2.9		dialog	-	supported	
Section 10.2.2.10		stream	-	supported	

Table 11. MSML Audit Connection Package Support

RFC 5707 Ref	Element Name	State Parameters	Attribute Name	Level of Support	Comments
Section 10.3	<audit>	Prefix: “audit.conn”	-	supported	Based on Table 9 Audit Core framework.
Section 10.3.1		sipdialog	-	supported	
		sipdialog.localseq	-	supported	
		sipdialog.remoteseq	-	supported	
		sipdialog.localURI	-	supported	
		sipdialog.remoteURI	-	supported	
		sipdialog.remotetarget	-	supported	
		sipdialog.routeset	-	supported	
		localsdp	-	supported	
		remotesdp	-	supported	
		dialog	-	supported	
		stream	-	supported	
		Child Elements			

Table 11. MSML Audit Connection Package Support

RFC 5707 Ref	Element Name	State Parameters	Attribute Name	Level of Support	Comments
Section 10.3.2.1	<auditresult>	sipdialog	callid	supported	Based on Table 9 Audit Core framework.
			localtag	supported	
			remotetag	supported	
Section 10.3.2.2		sipdialog.localseq	-	supported	
Section 10.3.2.3		sipdialog.remoteseq	-	supported	
Section 10.3.2.4		sipdialog.localURI	-	supported	
Section 10.3.2.5		sipdialog.remoteURI	-	supported	
Section 10.3.2.6		sipdialog.remotetarget	-	supported	
Section 10.3.2.7		sipdialog.routeset	-	supported	
Section 10.3.2.8		localsdp	-	supported	
Section 10.3.2.9		remotesdp	-	supported	
Section 10.3.2.10		dialog	-	supported	
Section 10.3.2.11		stream	-	supported	

Table 12. MSML Audit Dialog Package Support

RFC 5707 Ref	Element Name	State Parameters	Attribute Name	Level of Support	Comments
Section 10.4	<audit>	Prefix: “audit.conf.dialog” or “audit.conn.dialog	-	supported	Based on Table 9 Audit Core framework -Prefix selection depends on context of the stream state queried.
Section 10.4.1		dialog	-	supported	
		dialog.duration	-	supported	
		dialog.primitive	-	supported	
		dialog.controller	-	supported	
		Child Elements			
Section 10.4.2	<auditresult>	dialog	src	supported	Based on Table 9 Audit Core framework.
			type	supported	
			name	supported	
Section 10.4.2.1		dialog.duration	-	supported	
Section 10.4.2.2		dialog.primitive	-	supported	
Section 10.4.2.3		dialog.controller	-	supported	

Table 13. MSML Audit Stream Package Support

RFC 5707 Ref	Element Name	State Parameters	Attribute Name	Level of Support	Comments
Section 10.5	<audit>	Prefix: “audit.conf.dialog” or “audit.conn.dialog	-	supported	Based on Table 9 Audit Core framework -Prefix selection depends on context of the stream state queried.
Section 10.5.1		stream	-	supported	
		stream.clamp	-	supported	
		stream.gain	-	supported	
		stream.visual	-	not supported	
Section 10.5.2	<auditresult>	stream	joinwith	supported	Based on Table 9 Audit Core framework.
			media	supported	
			dir	supported	
			compressed	not supported	
			display	not supported	
			override	not supported	
			preferred	not supported	
Section 10.5.2.1		stream.clamp	-	supported	
Section 10.5.2.2		stream.gain	-	supported.	
Section 10.5.2.3		stream.visual		not supported	

3.3 MSML Support Details

The following sections provide details about MSML support.

3.3.1 MSML Core Package Support

This section will be provided in a future release.

3.3.2 MSML Conference Core Package Support

The following pages provide details about the MSML Conference Core Package elements listed below.

<asn>
: controls conference active speaker notification

<audiomix>
: specifies the properties of the conferencing audio mix

<clamp>
: filters tones and/or DTMF digits from an audio stream

<createconference>
: allocates and configures the media mixing resources for conferences

<destroyconference>
: deletes mixers or the entire conference.

<gain>
: specifies the gain characteristics applied to an audio stream, including the ability to mute and un-mute the stream

<join>
: creates one or more streams between two independent objects

<modifyconference>
: modifies the properties of an audio mix or the presentation of a video mix of a conference

<modifystream>
: modifies properties of an existing stream

<n-loudest>
: defines the number of participants that will be included in the conference mix based upon their audio energy

<region>
: defines video panes (or tiles) that are used to visually display participants of a video conference

<root>
: specifies the root window in which the video mix will be displayed

<selector>
specifies methods and associated parameters for automatic selection and displaying of video within a region or the root window of a conference

<stream>
manipulates and/or specifies properties of specific streams

<unjoin>
removes one or more streams between two independent objects

<videolayout>
specifies the properties of the conferencing video mix

<asn>

Parent:	<audiomix>
Child Elements:	none

Description

The **<asn>** element is a child of the <audiomix> element and may be used when creating or modifying a conference. It enables/disables notification of active speakers. Active speakers are notified using the <event> element with an event name of "msml.conf.asn". The namelist of the event consists of the set of active speakers. The name of each item is the string "speaker" with a value of the connection identifier for the connection.

Attributes

Attributes	Description
ri	Mandatory. Specifies the minimum reporting interval which defines the minimum duration of time that must pass before changes to active speakers will be reported. A value of zero disables active speaker notification. The minimum reporting interval may be set from 500 ms to 120 seconds. Values may be specified as milliseconds (i.e. 500ms), seconds (i.e. 2s), or minutes (i.e. 2m). Values specified without units will be interpreted as milliseconds. Values specified as milliseconds that are not multiples of 10ms will be rounded up to the nearest 10ms divisible value. Specifying values outside of the supported time interval range for “ri”, or values that are invalid in any other way, such as unsupported units (i.e. 10x), will result in an error response code of 410, invalid attribute value, being returned.

Events

msml.conf.asn

This is the active speaker notification event that will be generated by the media server. An example of an active speaker notification is as follows:

```
<event name="msml.conf.asn" id="conf:example">
               <name>speaker</name>
                <value>conn:hd93tg5hdf</value>
                <name>speaker</name>
                <value>conn:w8cn59vei7</value>
                <name>speaker</name>
               <value>conn:p78fnh6sek47fg</value>
</event>.
```

Shadow Variables

None.

Examples

The following examples illustrate the usage of the **<asn>** element:

- Section 7.1.1, “Creating a basic audio conference with <asn> and <n-loudest>”, on page 95.
- Section 7.1.2, “Modifying a basic audio conference with <asn> and <n-loudest>”, on page 95.

<audiomix>

Parent:	<createconference>, <modifyconference>, <destroyconference>
Child Elements:	<n-loudest>, <asn>

Description

The **<audiomix>** element specifies the properties of the conferencing audio mix. The properties of the overall audio mix are specified using the <audiomix> element and child elements <n-loudest> and/or <asn>.

Attributes

Attributes	Description
id	Optional. Specifies the identifier of the audio mix.

Events

None.

Shadow Variables

None.

Examples

The following examples illustrate the usage of the **<audiomix>** element:

- Section 7.1.1, “Creating a basic audio conference with <asn> and <n-loudest>”, on page 95.
- Section 7.1.2, “Modifying a basic audio conference with <asn> and <n-loudest>”, on page 95.

<clamp>

Parent: <stream>
Child Elements: None

Description

The **<clamp>** element is used to filter tones and/or DTMF digits from an audio stream and is support for audio streams flowing from a network connection object towards a conferencing object.

Attributes

Attributes	Description
dtmf	Mandatory: This attribute is used to enable DTMF tone clamping. A value of “true” enables DTMF tone clamping. A value of “false” disables DTMF tone clamping. The default value is “true”.
tone	Mandatory: This attribute is used to enable tone clamping and is not supported. Must be set to “false”.

Events

None.

Shadow Variables

None.

Examples

None.

<createconference>

Parent: <msml>
Child Elements: <audiomix>, <videolayout>

Description

The **<createconference>** element is used to allocate and configure the media mixing resources for conferences. A description of the properties for each type of media mix required for the conference is defined within the content of the **<createconference>** element. Mixer descriptions are described in Audio Mix and Video Layout sections. When no mixer descriptions are specified, the default behavior is equivalent to inclusion of a single<audiomix>.

Clients can request that a media server automatically delete a conference when a specified condition occurs by using the "deletewhen" attribute.

Attributes

Attributes	Description
name	Optional. Specifies the instance name (identifier) of the conference. If the attribute is not present, the media server assigns a globally unique name for the conference. If the attribute is present but the name is already in use, an error (432) will result and MSML document execution will stop. Events that the conference generates will use this name as the value of their "id" attribute.
deletewhen	Optional. Defines whether a media server should automatically delete the conference. Possible values are "nomedia", "nocontrol", and "never". Default is "nomedia". •A value of "nomedia" indicates that the conference MUST be deleted when no participants remain in the conference. When this occurs, an "msml.conf.nomedia" event notification is sent to the MSML client. •A value of "nocontrol" indicates that the conference MUST be deleted when the SIP [n1] dialog that carries the <createconference> element is terminated. When this occurs, the media server terminates all conference participant dialogs by sending a BYE for their associated SIP dialog. •A value of "never" leaves the ability to delete a conference under the control of the MSML client.
term	Optional. Possible values are "true", and "false". Default is "true". •A value of “true” indicates that the media server MUST send a BYE request on all SIP dialogs still associated with the conference when the conference is deleted. •For a value of “false”, SIP dialogs associated with the conference will not be automatically deleted by the media server when the conference is deleted.
mark	Optional. A token that MAY be used to identify execution progress in the case of errors. The value of the mark attribute from the last successfully executed MSML element is returned in an error response. Therefore, the value of all mark attributes within an MSML document should be unique.

Events

None.

Shadow Variables

None.

Examples

The following examples illustrate the usage of the **<createconference>** element:

- Section 7.1.1, “Creating a basic audio conference with <asn> and <n-loudest>”, on page 95.
- Section 7.2.1, “Creating a four party layout video conference”, on page 101.
- Section 7.2.5, “Single party layout video conference using a <selector> element”, on page 105.

<destroyconference>

Parent: <msml>
Child Elements: none

Description

The **<destroyconference>** element is used to delete mixers or to delete the entire conference and all state and shared resources. When a conference is destroyed, SIP dialogs for any remaining participants are maintained or removed based on the value of the "term" attribute when the conference was created. When there is no element content, **<destroyconference>** deletes the entire conference.

Attributes

Attributes	Description
id	Mandatory. This attribute specifies the identifier of the conference to be destroyed or to have mixers removed.
mark	Optional. A token that MAY be used to identify execution progress in the case of errors. The value of the mark attribute from the last successfully executed MSML element is returned in an error response. Therefore, the value of all mark attributes within an MSML document should be unique.

Events

None.

Shadow Variables

None.

Examples

The following example illustrates the usage of the **<destroyconference>** element:

- Section 7.1.9, “Destroying a conference”, on page 101.

<gain>

Parent: <stream>
Child Elements: None

Description

The **<gain>** element specifies the gain characteristics applied to an audio stream, including the ability to mute and un-mute the stream. It may be used to adjust the volume of an audio media stream and it may be set to apply a specific gain amount via the “amt’ attribute or to automatically adjust the gain to a configured target level via the “agc’ attribute. It also provides the ability to mute and un-mute the audio stream also using “amt’ attribute.

The **<gain>** element is supported for audio streams flowing between network connection objects and conferencing objects as follows:

- For audio streams flowing from a network connection object towards a conferencing object
  Setting: amt="n" (where n is a supported integer value for the amt attribute)
  Un-mutes the audio stream flowing into the conference, if previously muted, and sets the gain to the specified value.

  Setting: amt="mute"
  This mutes the audio stream flowing into the conference.

  Setting: amt="unmute"
  Un-mutes the audio stream flowing into the conference.
- For audio streams flowing from a conferencing object towards a network connection object
  Setting: amt="mute"
  This mutes the audio stream flowing out of the conference.

  Setting: amt="unmute"
  Un-mutes the audio stream flowing out of the conference.

  Setting: amt="0"
  Un-mutes the audio stream flowing out of the conference.

  Setting: amt= (any other value other than "mute", “unmute”, or "0")
  For this stream, these values for the amt attribute are invalid and not supported.
- For audio streams flowing between two network connection objects
  Setting: amt="n" (where n is a supported integer value for the amt attribute)
  Sets the gain to the specified value.

  Setting: amt="mute" or amt=“unmute”
  For this stream, these values for the amt attribute are invalid and not supported.

Attributes

Attributes	Description
amt	This attribute can be used to specify a gain to apply to the stream. It can also be used to mute or un-mute the stream. Values supported include integers from -32 to 32 representing a gain in dB to apply to the stream. Also supported are the values “mute” and “un-mute”. •The amt attribute is supported for audio streams flowing from a network connection object towards a conferencing object, for audio streams flowing from a conferencing object towards a network connection object, and for audio streams flowing between two network connection objects. Also see the limitations outlined in the description section for the <gain> element.
agc	This attribute specifies whether automatic gain control should be enabled or disabled. Supported values are “true” and “false”. Default is “false”. •The agc attribute is supported for audio streams flowing from a network connection object towards a conferencing object. Also see the limitations outlined in the description section for the <gain> element.

Events

None.

Shadow Variables

None.

Examples

The following examples illustrate the usage of the **<gain>** element:

- Section 7.1.5, “Muting an audio stream flowing into a conference”, on page 97.
- Section 7.1.6, “Un-muting an audio stream flowing into a conference”, on page 97.

<join>

Parent: <msml>
Child Elements: <stream>

Description

The **<join>** element is used to create one or more streams between two independent objects identified by the id1 and id2 attributes. Streams may be audio or video and may be bidirectional or unidirectional. A bidirectional stream is implicitly composed of two unidirectional streams that can be manipulated independently. The streams to be established are specified by<stream> child elements as the content of **<join>**.

Without any content, **<join>** by default establishes a bidirectional audio stream. When only a stream of a single type has previously been created between two objects, or when only a unidirectional stream exists, **<join>** can be used to add a stream of another media type or make the stream bidirectional by including the necessary <stream> elements. Bidirectional streams are made unidirectional by using <unjoin> to remove the unidirectional stream for the direction that is no longer required.

In addition to defining the media type and direction of streams, <stream> elements are also used to establish the properties of streams, such as gain, voice masking, or tone clamping of audio streams, or labels and other visual characteristics of video streams. Properties are often defined asymmetrically for a single direction of a stream. Creating a bidirectional stream requires two <stream> elements within the **<join>**, one for each direction, if one direction is to have different properties from the other direction.

Attributes

Attributes	Description
Id1	Mandatory. This attribute specifies an identifier of either a connection or conference. Wildcards do not apply for join operations.
Id2	Mandatory. This attribute specifies an identifier of either a connection or conference. Wildcards do not apply for join operations.
mark	Optional. A token that MAY be used to identify execution progress in the case of errors. The value of the mark attribute from the last successfully executed MSML element is returned in an error response. Therefore, the value of all mark attributes within an MSML document should be unique.

Events

None.

Shadow Variables

None.

Examples

The following examples illustrate the usage of the **<join>** element:

- Section 7.1.3, “Joining six parties to an audio conference”, on page 96.
- Section 7.1.4, “Call center coach-pupil conference”, on page 96.
- Section 7.1.7, “Un-joining streams using wildcards”, on page 98.
- Section 7.1.8, “Continuous digit collection on a conference participant”, on page 99
- Section 7.2.1, “Creating a four party layout video conference”, on page 101.
- Section 7.2.2, “Expanding a four party layout to a six party layout video conference”, on page 102.
- Section 7.2.5, “Single party layout video conference using a <selector> element”, on page 105.
- Section 7.2.6, “Sequencing parties through regions in a video conference layout”, on page 105.

<modifyconference>

Parent: <msml>
Child Elements: <audiomix>, <videolayout>

Description

The **<modifyconference>** element is used to modify the properties of an audio mix or the presentation of a video mix of a conference. All of the properties of an audio mix or the presentation of a video mix may be changed during the life of a conference using the **<modifyconference>** element. Changes to an audio mix are requested by including an <audiomix> element as a child of **<modifyconference>**. This may also be used to add an audio mixer to the conference if none was previously allocated.

Changes to a video presentation are requested by including a <videolayout> element as a child of **<modifyconference>**. Similar to an audio mixer, this may be used to add a video mixer if none was previously allocated. Mixers are removed by including a mixer description element within <destroyconference>.

Features and presentation aspects are enabled/added or modified by including the element(s) that define the feature or presentation aspect within a mixer description. The complete specification of the element must be included just as it would be included when the conference is created. The new definition completely replaces any previous definition that existed. Only things that are defined by elements included in the mixer descriptions are affected. Any existing configuration aspects of a conference, which are not specified within the **<modifyconference>** element, maintain their current state in the media server.

Attributes

Attributes	Description
id	Mandatory. Specifies the identifier of the conference to be modified.
mark	Optional. A token that MAY be used to identify execution progress in the case of errors. The value of the mark attribute from the last successfully executed MSML element is returned in an error response. Therefore, the value of all mark attributes within an MSML document should be unique.

Events

None.

Shadow Variables

None.

Examples

The following examples illustrate the usage of the **<modifyconference>** element:

- Section 7.1.2, “Modifying a basic audio conference with <asn> and <n-loudest>”, on page 95.

- Section 7.2.2, “Expanding a four party layout to a six party layout video conference”, on page 102.
- Section 7.2.3, “Contracting a six party layout to a four party layout video conference”, on page 103.
- Section 7.2.4, “Layered regions in a video conference”, on page 104.

<modifystream>

Parent: <msml>
Child Elements: <stream>

Description

The **<modifystream>** element is used to modify properties of an existing stream. Media streams can have different properties such as the gain for an audio stream or a visual label for a video stream. These properties are specified as the content of <stream> elements. The **<modifystream>** element is used to change the properties of a stream by including one or more <stream> elements that are to have their properties changed.

Stream properties are set as specified by the element <stream> as a child element of the **<modifystream>** element. Any properties not included in the **<stream>** element when modifying a stream will remain unchanged. Setting a property for only one direction of a bidirectional stream will NOT affect the other direction. The direction of streams can be changed by issuing an <unjoin> followed by a <join>. Any streams that exist between the two objects that are not included within **<modifystream>** will not be affected.

Attributes

Attributes	Description
Id1	Mandatory. This attribute specifies an identifier of either a connection or conference. Wildcards do not apply for join operations.
Id2	Mandatory. This attribute specifies an identifier of either a connection or conference. Wildcards do not apply for join operations.
mark	Optional. A token that MAY be used to identify execution progress in the case of errors. The value of the mark attribute from the last successfully executed MSML element is returned in an error response. Therefore, the value of all mark attributes within an MSML document should be unique.

Events

None.

Shadow Variables

None.

Examples

The following examples illustrate the usage of the **<modifystream>** element:

- Section 7.1.5, “Muting an audio stream flowing into a conference”, on page 97.
- Section 7.1.6, “Un-muting an audio stream flowing into a conference”, on page 97.
- Section 7.2.6, “Sequencing parties through regions in a video conference layout”, on page 105.

<n-loudest>

Parent: <audiomix>
Child Elements: none

Description

The **<n-loudest>** element defines the number of participants that will be included in the conference mix based upon their audio energy. It specifies the maximum number of conference participants that will be summed as part of the audio mix at any time. Participants to be mixed are determined by audio energy levels.

When the **<n-loudest>** element has not been included when creating, nor when modifying a conference, the maximum number of conference participants that will be summed as part of the audio mix at any time will be equivalent to the default for the media server. This default may be set via configuration options provided for the media server.

Attributes

Attributes	Description
n	Mandatory. The attribute “n” specifies the number of participants as mentioned above. Supported values are integers from 2 to 10.

Events

None.

Shadow Variables

None.

Examples

The following examples illustrate the usage of the **<n-loudest>** element:

- Section 7.1.1, “Creating a basic audio conference with <asn> and <n-loudest>”, on page 95.
- Section 7.1.2, “Modifying a basic audio conference with <asn> and <n-loudest>”, on page 95.

<region>

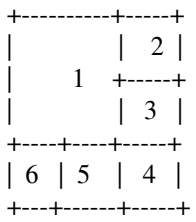
Parent: <videolayout>
Child Elements: None

Description

The **<region>** element is used to define video panes (or tiles) that are used to display participant video streams in a video conference. Regions are rendered on top of the root window. Up to 10 regions may be defined for each conference.

The location of the top left corner of a region is specified using the position attributes "top" and "left" and is defined relative to the top left corner of the root window. The size of a region is specified using the "relativesize" attribute and is defined relative to the size of the root window.

An example of a video layout with six regions is:



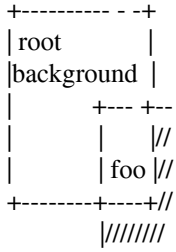

```
<videolayout type="text/msml-basic-layout">
         <root size="CIF"/>
         <region id="1" left="0" top="0" relativesize="2/3"/>
         <region id="2" left="67%" top="0" relativesize="1/3"/>
         <region id="3" left="67%" top="33%" relativesize="1/3">
         <region id="4" left="67%" top="67%" relativesize="1/3"/>
         <region id="5" left="33%" top="67%" relativesize="1/3"/>
         <region id="6" left="0" top="67%" relativesize="1/3"/>
</videolayout>
```

Portions of regions that extend beyond the root window will be cropped.

For example, a layout specified as:

```
<videolayout>
       <root size="CIF"/>
      <region id="foo" left="50%" top="50%" relativesize="2/3"/>
</videolayout>
```

would appear similar to:



The area of the root window covered by a region is a function of the region's position and its size. When areas of different regions overlap, they are layered in order of their "priority" attribute.

The region with the highest value for the "priority" attribute is below all other regions and will be hidden by overlapping regions. The region with the lowest non-zero value for the "priority" attribute is on top of all other regions and will not be hidden by overlapping regions. The priority attribute may be assigned values between 0 and 1. Note that a value of "0" is currently not supported. According to RFC5707, a value of "0" disables the region, freeing any resources associated with the region, and unjoining any video stream displayed in the region. Since a value of "0" is not supported, these steps must be done explicitly. A region can be made invisible with the relativesize attribute set to a value of "0" and can be modified using **<modifyconference>**. The region itself, once create will not be destroyed until a **<destroyconference>** is invoked. The "priority" attribute set to a value of "0' will also currently make a region invisible but this behavior will be modified in the future when the behavior specified in the RFC for a value of "0" is supported.

Regions that do not specify a priority will be assigned a priority by a media server when a conference is created. The first region within the **<videolayout>** element that does not specify a priority will be assigned a priority of one, the second a priority of two, etc. In this way, all regions that do not explicitly specify a priority will be underneath all regions that do specify a priority. As well, within those regions that do not specify a priority, they will be layered from top to bottom, in the order they appear within the **<videolayout>** element.

For example, if a layout was specified as follows:

```
<videolayout>
     <root size="CIF"/>
     <region id="a" ... priority=".3" .../>
     <region id="b" ... />
     <region id="c" ... priority=".2" ...>
     <region id="d" ... />
</videolayout>
```

Then the regions would be layered, from top to bottom, c,a,b,d.

Attributes

Attributes	Description
id	Mandatory. This attribute specifies a name that is used to refer to the region. For example, reference to a region is required when modifying the characteristics of a region and when specifying which region a video stream will be displayed in. Note that once a region is created for a conference, that region will continue to exist for the life of the conference. Therefore only 10 regions with 10 unique ids can be used for regions of a conference. The region can be made invisible and it can be reused with different characteristics. But the region and its id will only be destroyed when the conference that it belongs to is destroyed or the video mix of the conference that it belongs to is destroyed (see **<destroyconference>**).
left	This attribute specifies the position of the left edge of the region as a relative offset from the left edge of the root window. Values may be expressed either as a percent (%) or fraction (x/y) of the horizontal dimension of the root window. Supported values, when expressed as a percent, range from 100.0000% to 100.0000%. Supported values, when expressed as a fraction, range from 100.0000/001.0000 to 100.0000/001.0000.
top	This attribute specifies the position of the top edge of the region as a relative offset from the top edge of the root window. Values may be expressed either as a percent (%) or fraction (x/y) of the horizontal dimension of the root window. Supported values, when expressed as a percent, range from 100.0000% to 100.0000%. Supported values, when expressed as a fraction, range from 100.0000/001.0000 to 100.0000/001.0000.
relativesize	This attribute specifies the size of the region relative to the root window. Since the size is specified relative to the root window, the aspect ratio of the region will be the same as the aspect ratio of the root window. Values may be expressed either as a percent (%) or fraction (x/y) of the root window. Supported values, when expressed as a percent, range from 000.0000% to 100.0000%. Supported values, when expressed as a fraction, range from 0 to 100.0000/001.0000. When the attribute relativesize is set to a value of “0”, the region will continue to exist but the region will become invisible.
priority	This attribute specifies a priority level determining how regions are visible when they overlap with other regions. Regions with lower priority levels will be layered on top of regions with higher priority levels. Supported values are 0.1 to 0.9 (0.1, 0.2, …, 0.9). If no value is specified, a priority value of “>1” is assigned dependent upon the order of creation. For regions with the same priority level, the last region created will be layered on top of previous regions created. Note that a value of “0” is currently not supported.

Events

None.

Shadow Variables

None.

Examples

The following examples illustrate the usage of the **<region>** element:

- Section 7.2.1, “Creating a four party layout video conference”, on page 101.

- Section 7.2.2, “Expanding a four party layout to a six party layout video conference”, on page 102.
- Section 7.2.3, “Contracting a six party layout to a four party layout video conference”, on page 103.
- Section 7.2.4, “Layered regions in a video conference”, on page 104.

<root>

Parent: <videolayout> <selector>
Child Elements: None

Description

The **<root>** element describes the root window or virtual screen in which the conference video mix will be displayed. Simple conferences can display participant video directly within the root window but more complex conferences will use regions for this purpose. Areas of the window, for this case, which are not used to display video, will show the root window background.

All video presentations require a root window. It MUST be present when a video mix is created and it cannot be deleted; however, its attributes MAY be changed using the <modifyconference> element.

Attributes

Attributes	Description
size	This attribute specifies the resolution of the root window. Supported values are: SQCIF, QCIF, CIF, QVGA, and VGA. The attribute is mandatory when creating the conference.

Events

None.

Shadow Variables

None.

Examples

The following examples illustrate the usage of the **<root>** element:

- Section 7.2.1, “Creating a four party layout video conference”, on page 101.
- Section 7.2.2, “Expanding a four party layout to a six party layout video conference”, on page 102.
- Section 7.2.3, “Contracting a six party layout to a four party layout video conference”, on page 103.
- Section 7.2.4, “Layered regions in a video conference”, on page 104.
- Section 7.2.5, “Single party layout video conference using a <selector> element”, on page 105.

<selector>

Parent: <videolayout>
Child Elements: <root>

Description

The **<selector>** element is used to define the selection criteria and its associated parameters when choosing one of several video streams to be automatically selected and displayed. The selection algorithm used to select the video stream is specified by the "method" attribute. Currently, “vas: Voice Activated Switching is the only supported method.

The **<selector>** element is supported for simple video conferences that display the video directly in the root window. For this case, the <root> element can be placed as a child of the **<selector>** element. Region elements MUST NOT be used in this case.

Attributes

Attributes	Description
id	Mandatory. This attribute specifies a name that is used to refer to the selector. For example, reference to a selector is required when specifying which selector region (or root) that a video stream will be displayed in.
method	The name of the method used to select the video stream. Supported values are "vas" (Voice Activated Switching).

Events

None.

Shadow Variables

None.

Examples

The following example illustrates the usage of the **<selector>** element:

- Section 7.2.5, “Single party layout video conference using a <selector> element”, on page 105.

<stream>

Parent: <join>, <unjoin>, <modifystream>
Child Elements: <gain>, <clamp>

Description

The **<stream>** element is used to manipulate and/or specify properties of specific individual streams. They may be included as a child element in any of the stream manipulation elements **<join>**, **<modifystream>**, or **<unjoin>**. The type of the stream, audio or video, is specified using a "media" attribute.

A bidirectional stream is identified when no direction attribute "dir" is present. A unidirectional stream is identified when a direction attribute is present. Additional properties via attributes and child elements may be specified as the content of **<stream>** elements when the element is used as a child of **<join>** or **<modifystream>**. Other than specifying “media” and “dir”, additional properties via attributes and child elements should not be specified when streams are removed using **<stream>** elements as a child of the **<unjoin>** element.

Attributes

Attributes	Description
media	Mandatory. Value must be set to "audio" or video".
dir	Optional. Value may be set to "from-id1" or "to-id1". These values are relative to the identifier attributes of the parent element.

Attributes for audio streams that are inputs to a conference

The attributes preferred and dlgc:conf_party_type are **<stream>** attributes specifically for audio streams that are formed when joining participants to a conference. These attributes MAY be used for an audio stream that is an input to a conference and MUST NOT be used for other streams.

Attributes	Description
preferred	Optional. Defines if the stream will always be mixed and audible to conference participants or whether it will need to contend for N-loudest mixing. •A value of "true" means that the stream will always be mixed. This means that party's input, providing its speech level is greater than zero, is always included in the output summation process along with the loudest remaining “Standard/Pupil” parties within the active talker limit defined for the conference. •A value of "false" means that the stream MAY contend for mixing into a conference when N-loudest mixing is enabled. Default is "false".
dlgc:conf_party_type	Optional. This is a Dialogic extension. This attribute specifies a conference party type. •A value of "coach" may be specified. Two selected parties can establish a private communication link within the overall conference. The coach is a private member of the conference and is only heard by the pupil. However, the pupil cannot speak privately with the coach. •A value of "pupil" may be specified. See “coach” above. •A value of "standard" may be specified. Default is "standard".

Attributes for video streams that are inputs to a conference

Attributes	Description
display	Optional. This attribute specifies the identifier of a video layout region or selector that is to be used to display the video stream.

Events

None.

Shadow Variables

None.

Examples

The following examples illustrate the usage of the **<stream>** element:

- Section 7.1.3, “Joining six parties to an audio conference”, on page 96.
- Section 7.1.4, “Call center coach-pupil conference”, on page 96.
- Section 7.1.5, “Muting an audio stream flowing into a conference”, on page 97.
- Section 7.1.6, “Un-muting an audio stream flowing into a conference”, on page 97.
- Section 7.1.7, “Un-joining streams using wildcards”, on page 98.
- Section 7.2.1, “Creating a four party layout video conference”, on page 101.
- Section 7.2.2, “Expanding a four party layout to a six party layout video conference”, on page 102.
- Section 7.2.5, “Single party layout video conference using a <selector> element”, on page 105.
- Section 7.2.6, “Sequencing parties through regions in a video conference layout”, on page 105.

<unjoin>

Parent: <msml>
Child Elements: <stream>

Description

The **<unjoin>** element is used to remove one or more existing media streams flowing between two independent objects identified by the id1 and id2 attributes. The **<unjoin>** element may also be used to remove streams flowing between a specific object and any other object by using wildcards in one of the identifier attributes.

In the absence of any child elements for the **<unjoin>** element, all media streams between the objects will be removed. Individual streams may be removed by specifying them using <stream> child elements, while the unspecified streams will not be removed. A bidirectional stream is changed to a unidirectional stream by unjoining the direction that is no longer required, using the **<unjoin>** element. Operator elements MUST NOT be specified within **<stream>** elements when streams are being removed using the **<unjoin>** element. Any specified stream operators included will be ignored.

Attributes

Attributes	Description
id1	Mandatory. This attribute specifies an identifier of either a connection or conference. Wildcards do not apply for join operations.
id2	Mandatory. This attribute specifies an identifier of either a connection or conference. Wildcards do not apply for join operations.
mark	Optional. A token that MAY be used to identify execution progress in the case of errors. The value of the mark attribute from the last successfully executed MSML element is returned in an error response. Therefore, the value of all mark attributes within an MSML document should be unique.

Events

None.

Shadow Variables

None.

Examples

The following examples illustrate the usage of the **<unjoin>** element:

- Section 7.1.7, “Un-joining streams using wildcards”, on page 98.
- Section 7.2.3, “Contracting a six party layout to a four party layout video conference”, on page 103.

<videolayout>

Parent: <createconference>, <modifyconference>
Child Elements: <root>, <region>, <selector>

Description

A video layout is specified using the **<videolayout>** element. It is used as a container to hold elements that describe all of the properties of a video mix. The parameters of the window that displays the video mix are defined by the <root> element. When the video mix in composed of multiple panes, the location and characteristics of the panes are defined by one or more <region> elements. A <region> element is not required when only a single video stream is displayed at one time and none of the visual attributes of regions are required.

Attributes

Attributes	Description
type	Optional. When specified, type must equal “text/msml-basic-layout”.
id	Optional. An optional identifier for the video layout.

Events

None.

Shadow Variables

None.

Examples

The following examples illustrate the usage of the **<videolayout>** element:

- Section 7.2.1, “Creating a four party layout video conference”, on page 101.
- Section 7.2.2, “Expanding a four party layout to a six party layout video conference”, on page 102.
- Section 7.2.3, “Contracting a six party layout to a four party layout video conference”, on page 103.
- Section 7.2.4, “Layered regions in a video conference”, on page 104.
- Section 7.2.5, “Single party layout video conference using a <selector> element”, on page 105.

3.3.3 MSML Dialog Package Support

This section will be provided in a future release.

3.3.4 MSML Audit Package Support

This section will be provided in a future release.

Deviations from IETF RFC 4

The version of the MSML Media Server Software described in this manual is based on the IETF standard RFC 5707.

The following is a list of deviations from the IETF RFC:

- Nested groups are not supported.
- Only the “parallel” **topology** for <**group**> elements is supported.
- The RFC calls for the <**play**> element to be a child of a <**record**> or <**dtmf**> element. This is not supported.
- Wildcard IDs for the <**modifystream**> element are not supported.
- The <**video**> element is not supported as a direct child of the <**play**> element. The <**audio**> and <**video**> elements must be child elements of the <**media**> element to play an audio-visual recording.
- The **barge**, **cleardb**, and **offset** attributes of the <**play**> element have no effect when playing a video.
- The **play.amt** shadow variable of the <**play**> element has no effect for video.
- The **record.len** shadow variable of the <**record**> element has no effect for video.
- When recording an audio item, the **dest** attribute must be used.
- When recording an audio-visual item, the **audiodest** and **videodest** attributes must be used.
- Audio-visual recordings are currently recorded into two separate files.
- Audio-visual playback is supported via two separate files and must be defined in an <**audio**> and <**video**> element.
- The **format** attribute of the <**pattern**> element supports the “moml+digits” format only. The “mgcp” and “megaco” formats are not supported.

Feature Details 5

This chapter describes the features supported by the current version of the MSML Media Server Software in detail and provides information on how to use these features. Topics include:

- SIP Session Timer 77
- SIP re-INVITE. 80
- HTTP Play and Record 82

Note: A feature may not be supported on all platforms. For more information, see Table 1, "High-Level Feature Summary", on page 24 in Chapter 3, "Feature and Protocol Package Support" .

5.1 SIP Session Timer

The session timer feature provides a keepalive mechanism for the Session Initiation Protocol (SIP) to determine whether a session is still active using an extension defined in RFC 4028.

5.1.1 Feature Description

Session timer can be enabled by the application server on a per call basis through a SIP INVITE request to the media server. The support is disabled in the media server by default. The application server can also set the timer refresh interval on a per call basis.

For every SIP session, the session timer is active if requested by the application server. If the session timer is active, the application server sends a refresh message to the media server at regular intervals. The media server must respond to the refresh message, which is a SIP UPDATE request. The refresh response is a 200 OK.

If the media server does not receive the refresh message during the refresh interval, the associated SIP session is cleared by the media server. Any active dialog associated with the SIP session is terminated.

If the application server does not receive the refresh response from the media server during the refresh interval, the associated SIP session is cleared by the application server.

5.1.2 Enabling SIP Session Timer

The session timer support in the media server is disabled by default. It is enabled when the application server requests session timers through the SIP INVITE request.

This section illustrates a sample scenario where the application server (AS) requests session timer support from the media server (MS).

AS: SIP INVITE request for session timer support

The application server sends a SIP INVITE request to the media server requesting session timer support (see text in bold).

```
INVITE sip:192.168.1.227:5060 SIP/2.0
From: <sip:192.168.1.210:5060>;tag=2cd4db0-0-13c4-4086a-2f8ff578-4086a
To: <sip:192.168.1.227:5060>
Call-ID: 2ce18c8-0-13c4-4086a-4de0d2d3-4086a@192.168.1.210
CSeq: 1 INVITE
Via: SIP/2.0/UDP 192.168.100.160:5060;branch= 4086a-fc0df70-31ac7aa1
Max-Forwards: 70
Supported: timer
Contact: <sip:192.168.1.210:5060>
Session-Expires: 90;refresher=uac
Min-SE: 90
Allow: INVITE, CANCEL, ACK, BYE, OPTIONS, UPDATE
Content-Type: application/SDP
Content-Length: 239

v=0
o=- 1032127810 1032127810 IN IP4 192.168.1.215
s=Polycom IP Phone
c=IN IP4 192.168.1.215
t=0 0
m=audio 2240 RTP/AVP 0 8 18 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:101 telephone-event/8000
```

MS: SIP response to request for session timer support

The media server responds with a 200 OK message and agrees to use session timers (see text in bold).

```
SIP/2.0 200 OK
From: <sip:192.168.1.210:5060>;tag=2cd4db0-0-13c4-4086a-2f8ff578-4086a
To: <sip:192.168.1.227:5060>;tag=31d86c8-0-13c4-42a04-4f95866a-42a04
Call-ID: 2ce18c8-0-13c4-4086a-4de0d2d3-4086a@192.168.1.210
CSeq: 1 INVITE
Via: SIP/2.0/UDP 192.168.100.160:5060;received=192.168.1.210;branch=
       4086a-fc0df70-31ac7aa1
Supported: timer
Require: timer
Contact: <sip:192.168.1.227:5060>
Session-Expires: 90;refresher=uac
Content-Type: application/SDP
Content-Length: 268
v=0
o=MediaServer 1180967421 1180967421 IN IP4 192.168.1.227
s=-
i=Media Server Version 1,0,0,75
t=0 0
m=audio 49188 RTP/AVP 0 101
c=IN IP4 192.168.1.227
a=rtpmap:0 PCMU/8000/1
a=rtpmap:101 telephone-event/8000
a=fmtp:0 ptime=20
a=fmtp:101 0-15
a=sendrecv
```

At this point, the call referenced above is active and session timer support is active.

AS: SIP UPDATE request

Once the SIP session is established with session timers enabled, the application server sends a SIP UPDATE request to the media server at regular intervals, as specified in the initial INVITE. Once the media server receives the SIP UPDATE, it responds to the session timer refresh request with 200 OK.

```
UPDATE sip:192.168.1.227:5060 SIP/2.0
From: <sip:192.168.1.210:5060>;tag=2cd4db0-0-13c4-4086a-2f8ff578-4086a
To: <sip:192.168.1.227:5060>;tag=31d86c8-0-13c4-42a04-4f95866a-42a04
Call-ID: 2ce18c8-0-13c4-4086a-4de0d2d3-4086a@192.168.1.210
CSeq: 38 UPDATE
Via: SIP/2.0/UDP 192.168.100.160:5060;branch= 40cc0-fd1d1a6-29630a07
Max-Forwards: 70
Supported: timer
Contact: <sip:192.168.1.210:5060>
Session-Expires: 90;refresher=uac
Min-SE: 60
Allow: INVITE, CANCEL, ACK, BYE, OPTIONS, UPDATE
Content-Length: 0
```

MS: SIP response to SIP UPDATE request

The media server responds to the SIP UPDATE request from the application server with 200 OK.

If session timer support is enabled, the call remains active as long as the application server sends the SIP UPDATE request and the media server responds with 200 OK.

If the application server does not send the SIP UPDATE request after the session timer expires, or the media server does not respond with 200 OK, then the call is automatically terminated.

```
SIP/2.0 200 OK
From: <sip:192.168.1.210:5060>;tag=2cd4db0-0-13c4-4086a-2f8ff578-4086a
To: <sip:192.168.1.227:5060>;tag=31d86c8-0-13c4-42a04-4f95866a-42a04
Call-ID: 2ce18c8-0-13c4-4086a-4de0d2d3-4086a@192.168.1.210
CSeq: 38 UPDATE
Via: SIP/2.0/UDP 192.168.100.160:5060;received=192.168.1.210;branch=
      40cc0-fd1d1a6-29630a07
Supported: timer
Require: timer
Contact: <sip:192.168.1.227:5060>
Session-Expires: 90;refresher=uac
Content-Length: 0
```

5.1.3 Using SIP Session Timer with Dialogic® HMP Software

This section applies only to application servers that use Dialogic® Host Media Processing (HMP) Software as the call control mechanism.

To activate the session timer support in the media server, the application server must request session timer support in the SIP INVITE request.

The following procedure outlines the steps required to request session timer support from the media server using the Dialogic® Global Call API library in Dialogic® HMP Software.

1. Enable the session timer support on the virtual board on the application server.

```
m_virtBoard[0].E_SIP_SessionTimer_Enabled= ENUM_Enabled;
```

2. Set the default session timer refresh interval and minimum refresh interval as appropriate. The default session expires timer is set to 90 seconds below. If not set, it defaults to 1800 seconds.

```
m_virtBoard[0].SIP_SessionTimer_SessionExpires= 90;

m_virtBoard[0].SIP_SessionTimer_MinSE= 90;
```

3. Set the following IPPARM values on each SIP device.

```
gcParmBlk = NULL;
gc_util_insert_parm_val(&gcParmBlk,
        IPSET_SIP_SESSION_TIMER,
        IPPARM_REFRESH_METHOD,
        sizeof(UINT32),
        IP_REFRESH_UPDATE);
gc_util_insert_parm_val(&gcParmBlk,
        IPSET_SIP_SESSION_TIMER,
        IPPARM_REFRESHER_PREFERENCE,
        sizeof(UINT32),
        IP_REFRESHER_LOCAL);
gc_util_insert_parm_val(&gcParmBlk,
        IPSET_SIP_SESSION_TIMER,
        IPPARM_REFRESH_WITHOUT_REMOTE_SUPPORT,
        sizeof(UINT32),
        IP_REFRESH_WITHOUT_REMOTE_SUPPORT_DISABLE);
gc_util_insert_parm_val(&gcParmBlk,
        IPSET_SIP_SESSION_TIMER,
        IPPARM_REFRESH_WITHOUT_PREFERENCE,
        sizeof(UINT32),
        IP_REFRESH_WITHOUT_PREFERENCE_DISABLE);
gc_SetUserInfo(GCTGT_GCLIB_CHAN, m_DeviceHandle, gcParmBlk,
             GC_ALLCALLS);
gc_util_delete_parm_blk(gcParmBlk);
```

Note that the application server itself is not responsible for sending the session timer refresh requests. The underlying Dialogic® HMP Software manages interactions related to session timer requests and responses.

For more information about the Dialogic® Global Call API functions and parameters used for SIP session timer, see the SIP Session Timer topic in the Release Update for your corresponding software release. For more information about Global Call API for IP, see the Dialogic® Global Call IP Technology Guide.

5.2 SIP re-INVITE

SIP re-INVITE support is provided to handle media flow direction control changes, media capabilities renegotiation, and outbound media destination IP address changes.

5.2.1 Feature Description

The re-INVITE method is a general purpose mechanism that can be used to modify or update most of the properties of a dialog (the most notable exceptions being the header fields that are used to identify the message as a subsequent INVITE rather than a new INVITE) or the associated media session.

This initial implementation of re-INVITE does not support the following:

- Requests sent from the media server (MS). The media server will only receive re-INVITE requests and respond to them.
- Hold or Retrieve when the “inactive” attribute is used.
- Switching to Fax.

5.2.2 Media Stream Direction Support

This feature supports both full-duplex and half-duplex media streams. The direction of the media stream is based on the direction attribute supplied in the Session Description Protocol (SDP) using one of the following direction attributes, “sendrecv,” “sendonly,” or “recvonly.”

- If no direction attribute is supplied and a valid connection address “c=” is supplied, a full duplex media stream is established.
- If a connection address “c=” is set to 0.0.0.0, then a half duplex receive only media stream is established.
- The re-INVITE is rejected if the connection address “c=” is set to 0.0.0.0 and the direction attribute is “sendrecv” or “recvonly”, or if the direction attribute is set to “inactive”.

Support is provided for receiving an initial inbound INVITE with or without a valid SDP. In the case where the INVITE received does not contain an SDP, the media server (MS) will respond with an SDP offer in the 200 OK. The media flow direction is based on the answer SDP. When the inbound INVITE contains an SDP, the MS will respond with a valid SDP answer in the 200 OK and the media flow direction will be based on the offer SDP.

When an initial inbound INVITE is received with a valid SDP with a connection address “c=” value of 0.0.0.0, the MS responds with a valid SDP answer in the 200 OK. Media flow will be started as receive only.

A re-INVITE can be received any time after the call is established. Multiple re-INVITE requests can be received on the same SIP session, but prior requests must be processed and responded to before a new request can be received.

The media stream remains active based on the previous media stream capabilities and direction. A re-INVITE request with an invalid SDP will be rejected, but this has no effect on the current media stream.

The answer or offer SDP contains a valid connection address “c=”. If the connection address “c=” equals 0.0.0.0 and no direction attribute is set, then a direction attribute of “sendonly” is assumed. If a direction attribute other than “sendonly” is set, the re-INVITE is rejected. The answer or offer SDP direction attribute cannot be set to “inactive”.

5.3 HTTP Play and Record

The application server has the ability to store media recordings directly to an HTTP server. The application server can also play a recording and/or retrieve MSML dialog source directly from an HTTP server.

5.3.1 Feature Description

HTTP support enables the retrieval of MSML scripts directly from a web server for execution. It also allows the storage and retrieval of audio and video recordings to and from the HTTP server.

5.3.2 Requirements for HTTP Support

The MSML Media Server Software uses the HTTP GET and HTTP PUT commands to retrieve and store files respectively. When using MSML attributes that specify the “http://” scheme, it is important that the HTTP server support the HTTP GET and HTTP PUT commands. Typically, HTTP servers support the HTTP GET command, but when receiving HTTP PUT commands, some servers require server-side scripts to actually store files.

5.3.3 Dialog Execution

An external MSML dialog/script can be retrieved and executed from an HTTP server. Dialogs are a class of objects that represent automated participants. Dialogs are created and destroyed through MSML.

The “http://” scheme for the src attribute of an MSML <dialogstart> is supported. MSML dialog execution commences after the entire MSML body has been retrieved from the HTTP server. The HTTP GET functionality is used to retrieve the information from the web server.

5.3.4 Audio and Video Playback

Both audio and video files are retrievable from the HTTP server.

An audio file/prompt can start playing before the entire file has been downloaded from the HTTP server. Currently, audio playback support is only available when playing a VOX or WAV file using a DX device via user I/O.

If an audio file is being played in conjunction with a video file, playback will not start until the entire audio and video files are downloaded. The "http://" scheme is supported in the uri attribute of the MSML <audio> and <video> elements. The uri attribute identifies the location of the audio or video file. The HTTP GET functionality will be used to retrieve the information from the web server. Currently, only the Dialogic proprietary video format is supported.

5.3.5 Audio and Video Recording

MSML supports storing audio and video files to an HTTP server. Both are stored locally and uploaded to the HTTP server immediately after the recording has completed.

New audio files are stored in the location identified in the “http://” scheme for the dest attribute of the MSML <record> element. The audio portion of an audio / video recording is stored in the location specified at “http://” scheme for the audiodest attribute, while the location of the new video file is stored in the location identified at “http://” scheme for the videodest attribute of the MSML <record> element. The HTTP PUT functionality is used to send the information to the web server.

Sample Use Case 6

This chapter describes a simple application that demonstrates many of the capabilities provided by the current version of the MSML Media Server Software. Topics include:

- Use Case Description. 85
- MSML Control Syntax in Use Case 87

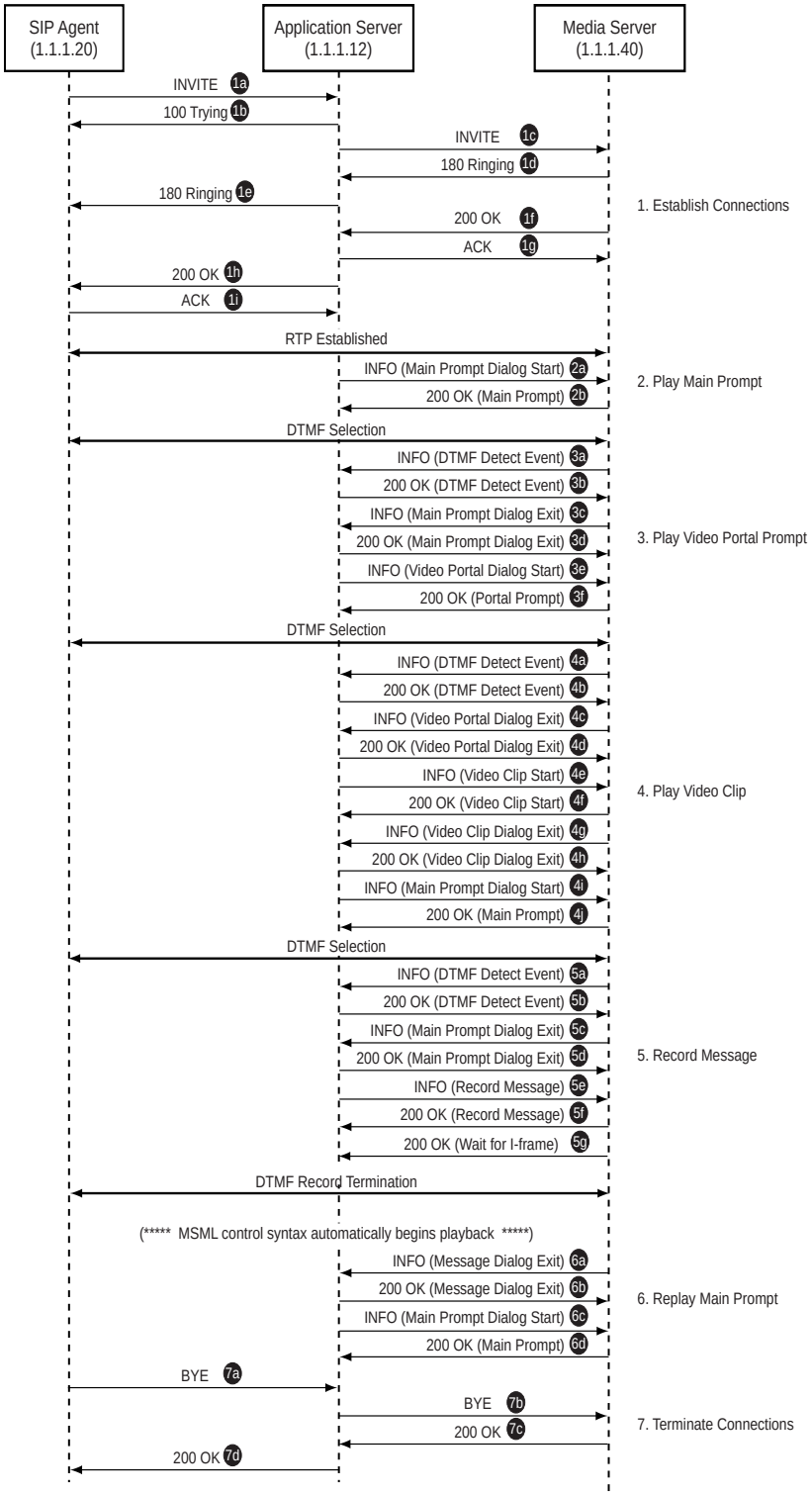
6.1 Use Case Description

In this application, the user is presented with options to play and record audio and video clips/messages. The application server (AS) communicates with the media server (MS) using the MSML Media Server Software to perform the selected operations. Figure 2 shows the exchange of SIP messages between the SIP client, AS, and MS to perform the functionality required. Many of the messages exchanged between the AS and MS include MSML control syntax that are interpreted and acted upon by the MSML Media Server Software.

The following sequence describes the high-level activities from the SIP client and MS perspectives:

1. The SIP client initiates a SIP dialog with the AS and a media session with the MS.
2. The MS plays the **Main Prompt** with options for the playing of prerecorded clips of News, Weather, Messages, Image of Your Daily Schedule, or the recording of an audio-visual message. The MS then waits on DTMF detection. The MS waits forever and never disconnects, unless a BYE is issued or unless AS timers set a limit on call length.
3. The SIP client makes a selection using DTMF. The selection is to display the **Video Portal Prompt**.
4. The MS plays the **Video Portal Prompt**.
5. The SIP client makes a selection using DTMF. The selection is to play a video.
6. The MS plays the selected video clip to completion.
7. The MS plays the **Main Prompt**.
8. The SIP client makes a selection using DTMF. The selection is to record, then play back, a video message.
9. The MS records the video message.
10. The SIP client stops the recording of the video message with any DTMF.
11. The MS starts the playback of the recorded video message (executed in MSML syntax).
12. The MS plays the recorded video message to completion.
13. The MS plays the **Main Prompt**.
14. The SIP client disconnects.
15. The MS disconnects.

Figure 2. Audio/Video Play/Record Scenario



6.2 MSML Control Syntax in Use Case

Figure 2 includes labels to identify the SIP messages exchanged among the SIP client, AS, and MS. For easier reference, the main steps are designated with numbers and subordinate steps are designated with lowercase letters. The following sections describe the steps (and subordinate steps) with particular emphasis on the MSML control syntax included in the exchanged messages. The main steps are:

- Establish Connections
- Play Main Prompt
- Play Video Portal Prompt
- Play Video Clip
- Record Message
- Replay Main Prompt
- Terminate Connections

Note: In the subsections following, the first SIP INFO message (in Section 6.2.2) is shown in its entirety to highlight the fact that the AS must set the "Content-Type" and "Content-Length" in the SIP header. For the remaining SIP messages, the SIP headers are not included since the focus is on the MSML control syntax.

6.2.1 Establish Connections

Steps 1a to 1i

These steps comprise standard SIP message exchange for the establishment of SIP dialogs between the SIP client and AS and between AS and MS and the establishment of a media session (RTP) between the SIP client and MS. It is over the RTP connection that the user responds to prompts using DTMF selections. There is no MSML control syntax involved in this message exchange.

However, one important piece of information received by the AS in **Step 1f** is the **network connection identifier** that is assigned by the MS. The identifier is the "tag" value included in the "To" header of the SIP 200 OK response to the initial INVITE sent by the AS. In this example, the "To" header is:

```
To:<sip:1.1.1.12>;tag=b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2
```

In MSML control syntax, the connection identifier is specified as "conn:<tag value>". In the sample control syntax in Section 6.2.2 to Section 6.2.6, the network connection identifier is shown in **bold** text.

6.2.2 Play Main Prompt

Step 2a

The AS sends the MS an INFO message to play the **Main Prompt** dialog. The complete INFO message shown below includes MSML control syntax:

```
INFO sip:AS@1.1.1.40:5060 SIP/2.0
Call-ID: ae11d01e-7350-462d-8a9e-169c79d7361a@1.1.1.20
From: "Administrator" <sip:WINDOWS-E6UOEQY>;tag=-1179327240.1.bababamaggmjjhbgpgjfogkj
To: <sip:1.1.1.12>;tag=b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2
CSeq: 3 INFO
Contact: sip:1.1.1.12:5060
Content-Type: text/xml;charset=UTF-8
Content-Length:709
Max-Forwards: 70
Via: SIP/2.0/UDP 1.1.1.12:5070;branch=z9hG4bK0101010CBADF00D00000109F178B4485

<?xml version="1.0" encoding="UTF-8" ?>
<msml version="1.1">
<dialogstart target="conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2"
type="application/msml+xml">
   <group topology="parallel" >
      <play>
         <media>
            <video uri="file://./av/main_menu.vid" format="video/raw:codecs=h263" />
            <audio uri="file://./av/main_menu.pcm" format="audio/pcm:codecs=mulaw"
                        audiosamplesize="8" audiosamplerate="8"/>
         </media>
      </play>
      <collect iterate="forever" cleardb="true">
         <detect>
            <send target="source" event="done" namelist="dtmf.digits" />
            <send target="group" event="terminate"/>
         </detect>
      </collect>
   </group>
</dialogstart>
</msml>
```

Step 2b

The MS sends a 200 OK response to indicate success. The 200 OK response includes the following MSML control syntax:

```
<msml version="1.1">
   <result response="200"/>
   <dialogid>conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2/dialog:10</dialogid>
</msml>
```

6.2.3 Play Video Portal Prompt

Step 3a

At this point, the SIP client makes a selection that is transmitted as DTMF to the MS. The selection is to display the **Video Portal Prompt** dialog. The MS sends the AS an INFO message that includes the following MSML control syntax describing the DTMF Detect event.

```
<msml version="1.1">
   <event name="done"
      id="conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2/dialog:10">
      <name>dtmf.digits</name><value>1</value>
   </event>
</msml>
```

Step 3b

The AS sends a 200 OK response to the MS to acknowledge successful receipt of the DTMF Detect event. The 200 OK response does not includes any MSML control syntax.

Step 3c

The MS sends the AS an INFO message with the following MSML control syntax to indicate that the **Main Prompt** dialog is exiting.

```
<msml version="1.1">
   <event name="msml.dialog.exit"
      id="conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2/dialog:10"/>
</msml>
```

Step 3d

The AS sends a 200 OK response to the MS to acknowledge **Main Prompt** dialog exit. The 200 OK response does not contain any MSML control syntax.

Step 3e

The AS sends the MS an INFO message to start playing the **Video Portal Prompt** dialog. The INFO message includes the following MSML control syntax:

```
<?xml version="1.0" encoding="UTF-8" ?>
<msml version="1.1">
<dialogstart target="conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2"
      type="application/msml+xml">
   <group topology="parallel" >
      <play>
         <media>
            <video uri="file://./av/vportal_menu.vid" format="video/raw:codecs=h263" />
            <audio uri="file://./av/vportal_menu.pcm" format="audio/pcm:codecs=mulaw"
                  audiosamplesize="8" audiosamplerate="8"/>
         </media>
      </play>
      <collect iterate="forever" cleardb="true">
         <detect>
            <send target="source" event="done" namelist="dtmf.digits dtmf.len dtmf.end
                  dtmf.last" />
            <send target="group" event="terminate"/>
         </detect>
      </collect>
   </group>
</dialogstart>
</msml>
```

Step 3f

The MS sends a 200 OK response to the AS to acknowledge the starting of the **Video Portal Prompt** dialog. The 200 OK response includes the following MSML control syntax:

```
<msml version="1.1">
   <result response="200"/>
   <dialogid>conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2/dialog:13</dialogid>
</msml>
```

6.2.4 Play Video Clip

Step 4a

At this point, the SIP client makes a selection that is transmitted as DTMF to the MS. The selection is to play a **Video Clip**. The MS sends the AS an INFO message that includes the following MSML control syntax describing the DTMF Detect event.

```
<msml version="1.1">
   <event name="done"
      id="conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2/dialog:13">
      <name>dtmf.digits</name><value>2</value>
      <name>dtmf.end</name><value>dtmf.detect</value>
      <name>dtmf.last</name><value>2</value>
      <name>dtmf.len</name><value>1</value>
   </event>
</msml>
```

Step 4b

The AS sends a 200 OK response to the MS to acknowledge successful receipt of the DTMF Detect event. The 200 OK response does not includes any MSML control syntax.

Step 4c

The MS sends the AS an INFO message with the following MSML control syntax to indicate that the **Video Portal Prompt** dialog is exiting (a response to the DTMF Detect event).

```
<msml version="1.1">
   <event name="msml.dialog.exit"
      id="conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2/dialog:13"/>
</msml>
```

Step 4d

The AS sends a 200 OK response to the MS to acknowledge **Video Portal Prompt** dialog exit. The 200 OK response does not contain any MSML control syntax.

Step 4e

The AS sends the MS an INFO message that includes the following MSML control syntax to start the **Video Clip** (a response to the DTMF Detect event).

```
<?xml version="1.0" encoding="UTF-8" ?>
<msml version="1.1">
<dialogstart target="conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2"
      type="application/msml+xml">
   <play>
      <media>
         <video uri="file://./av/clip2.vid" format="video/raw:codecs=h263" />
         <audio uri="file://./av/clip2.pcm" format="audio/pcm:codecs=mulaw"
            audiosamplesize="8" audiosamplerate="8"  />
      </media>
   </play>
</dialogstart>
</msml>
```

Step 4f

The MS sends a 200 OK response to the AS to acknowledge the starting of the **Video Clip**. The 200 OK response includes the following MSML control syntax:

```
<msml version="1.1">
   <result response="200"/>
   <dialogid>conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2/dialog:28</dialogid>
</msml>
```

Step 4g

The MS sends the AS an INFO message with the following MSML control syntax to indicate that the **Video Clip** dialog is exiting (a response to the DTMF Detect event).

```
<msml version="1.1">
   <event name="msml.dialog.exit"
      id="conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2/dialog:28"/>
</msml>
```

Step 4h

The AS sends a 200 OK response to the MS to acknowledge **Video Clip** dialog exit. The 200 OK response does not contain any MSML control syntax.

Step 4i

The AS sends the MS an INFO message to play the **Main Prompt** dialog. The INFO message includes the following MSML control syntax:

```
<?xml version="1.0" encoding="UTF-8" ?><msml version="1.1"><dialogstart
      target="conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2" type="application/msml+xml">
   <group topology="parallel" >
      <play>
         <media>
            <video uri="file://./av/main_menu.vid" format="video/raw:codecs=h263" />
            <audio uri="file://./av/main_menu.pcm" format="audio/pcm:codecs=mulaw"
               audiosamplesize="8" audiosamplerate="8"/>
         </media>
      </play>
      <collect iterate="forever" cleardb="true">
         <detect>
            <send target="source" event="done" namelist="dtmf.digits" />
            <send target="group" event="terminate"/>
         </detect>
      </collect>
   </group>
</dialogstart>
</msml>
```

Step 4j

The MS sends a 200 OK response to indicate success. The 200 OK response includes the following MSML control syntax:

```
<msml version="1.1">
   <result response="200"/>
   <dialogid>conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2/dialog:8</dialogid>
</msml>
```

6.2.5 Record Message

Step 5a

At this point, the SIP client makes a selection that is transmitted as DTMF to the MS. The selection is to **Record Message**. The MS sends the AS an INFO message that includes the following MSML control syntax describing the DTMF Detect event.

```
<msml version="1.1">
   <event name="done"
      id="conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2/dialog:8">
      <name>dtmf.digits</name><value>2</value>
</event>
</msml>
```

Step 5b

The AS sends a 200 OK response to the MS to acknowledge successful receipt of the DTMF Detect event. The 200 OK response does not includes any MSML control syntax.

Step 5c

The MS sends the AS an INFO message with the following MSML control syntax to indicate that the **Main Prompt** dialog is exiting (a response to the DTMF Detect event):

```
<msml version="1.1">
   <event name="msml.dialog.exit"
      id="conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2/dialog:8"/>
</msml>
```

Step 5d

The AS sends a 200 OK response to the MS to acknowledge **Main Prompt** dialog exit. The 200 OK response does not contain any MSML control syntax.

Step 5e

The AS sends the MS an INFO message that includes the following MSML control syntax to start the **Record Message** dialog (a response to the DTMF Detect event).

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
   <dialogstart target="conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2"
      type="application/msml+xml">
      <group topology="parallel">
         <record beep="true"    audiodest="file://./mytest.pcm"
               videodest="file://./mytest.vid"    format="video/raw;codecs=mulaw,h263"
               audiosamplerate="8" audiosamplesize="8">
            <recordexit>
               <send target="play" event="resume"/>
            </recordexit>
         </record>
         <play initial="suspend">
            <media>
               <audio uri="file://./mytest.pcm" format="audio/pcm;codecs=mulaw"
                     audiosamplerate="8" audiosamplesize="8" />
               <video uri="file://./mytest.vid" format="video/vid;codecs=h263"/>
            </media>
            <playexit>
               <send target="dtmf" event="terminate"/>
               <send target="record" event="terminate"/>
            </playexit>
         </play>
         <dtmf iterate="forever">
            <detect>
```

```
            <send target="record" event="terminate"/>
          </detect>
        </dtmf>
      </group>
    </dialogstart>
</msml>
```

Step 5f

The MS sends a 200 OK response to the AS to acknowledge the starting of the **Record Message** dialog. The 200 OK response includes the following MSML control syntax:

```
<msml version="1.1">
   <result response="200"/>
   <dialogid>conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2/dialog:30</dialogid>
</msml>
```

Step 5g

Once the MS is ready to record, it sends the AS an INFO message with the following MSML control syntax to indicate that the MS is waiting for an I-frame. The AS must forward this message to the remote SIP client.

```
<?xml version="1.0" encoding="utf-8" ?>
<media_control><vc_primitive><to_encoder><picture_fast_update></picture_fast_update></
 to_encoder></vc_primitive></media_control>
```

Recording starts once an I-frame is received.

If the MS does not get an I-frame within 5 seconds, another message with the same syntax is sent to the AS indicating that an I-frame timeout occurred. The AS must also forward this message to the remote SIP client. The MS will start recording without a valid I-frame.

6.2.6 Replay Main Prompt

Step 6a

At this point, when the SIP client sends any DTMF to the MS, the recording operation stops and playback begins automatically (as determined by the MSML control syntax in Step 5e). The MS also sends the AS an INFO message that includes the following MSML control syntax indicating a **Record Message** dialog exit event.

```
<msml version="1.1">
   <event name="msml.dialog.exit"
      id="conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2/dialog:30"/>
</msml>
```

Step 6b

The AS sends a 200 OK response to the MS to acknowledge successful receipt of the event. The 200 OK response does not includes any MSML control syntax.

Step 6c

When the playback of the recorded message is complete, the AS sends the MS an INFO message to play the **Main Prompt** dialog. The INFO message includes the following MSML control syntax:

```
<?xml version="1.0" encoding="UTF-8" ?><msml version="1.1">
<dialogstart target="conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2"
type="application/msml+xml">
   <group topology="parallel" >
      <play>
         <media>
            <video uri="file://./av/main_menu.vid" format="video/raw:codecs=h263" />
            <audio uri="file://./av/main_menu.pcm" format="audio/pcm:codecs=mulaw"
                  audiosamplesize="8" audiosamplerate="8"/>
         </media>
      </play>
      <collect iterate="forever" cleardb="true">
         <detect>
            <send target="source" event="done" namelist="dtmf.digits" />
            <send target="group" event="terminate"/>
         </detect>
      </collect>
   </group>
</dialogstart>
</msml>
```

Step 6d

The MS sends a 200 OK response to indicate success. The 200 OK response includes the following MSML control syntax:

```
<msml version="1.1">
   <result response="200"/>
   <dialogid>conn:b19d82a0-0-13c4-13cbd2-58c3964e-13cbd2/dialog:29</dialogid>
</msml>
```

6.2.7 Terminate Connections

Step 7a to 7d

These steps comprise standard SIP message exchanges for the termination of the SIP dialogs between the SIP client and the AS and between the AS and MS and the termination of the media session (RTP) between the SIP client and the MS.

MSML Script Examples 7

This chapter provides sample MSML scripts. Topics include:

- MSML Scripts for Audio Conferencing 95
- MSML Scripts for Video Conferencing 101

7.1 MSML Scripts for Audio Conferencing

The following sections provide examples of audio conferencing tasks.

7.1.1 Creating a basic audio conference with <asn> and <n-loudest>

The following example creates a basic audio conference with up to three active talkers included in the audio mix and active speaker notification with the minimum reporting interval set to 10 seconds.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.0">
   <createconference id="conf:1234" deletewhen="never">
       <audiomix id="mix1">
          <n-loudest n="3"/>
          <asn ri="10s"/>
       </audiomix>
   </createconference>
</msml>
```

7.1.2 Modifying a basic audio conference with <asn> and <n-loudest>

The following example modifies a basic audio conference to support up to five active talkers included in the audio mix and active speaker notification with the minimum reporting interval set to one second.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.0">
    <modifyconference id="conf:1234">
          <audiomix id="mix1">
             <n-loudest n="5"/>
             <asn ri="1s"/>
          </audiomix>
    </modifyconference>
</msml>
```

7.1.3 Joining six parties to an audio conference

The following example joins 12 parties to the audio conference created in example 7.1.1. The first party is the conference chair and is designated as a preferred party and will always be heard when speaking. The next five parties are also eligible to be heard when speaking based upon their audio energy levels. The last six parties are listen only parties.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
    <join id1="conn:0001" id2=" conf:1234">
        <stream media="audio" dir=”from-id1” preferred=”true”/>
        <stream media="audio" dir="to-id1"/>
    </join>
    <join id1="conn:0002" id2=" conf:1234"/>
    <join id1="conn:0003" id2=" conf:1234"/>
    <join id1="conn:0004" id2=" conf:1234"/>
    <join id1="conn:0005" id2=" conf:1234"/>
    <join id1="conn:0006" id2=" conf:1234"/>
    <join id1="conn:0007" id2=" conf:1234">
        <stream media="audio" dir="to-id1"/>
    </join>
    <join id1="conn:0008" id2=" conf:1234">
        <stream media="audio" dir="to-id1"/>
    </join>
    <join id1="conn:0009" id2=" conf:1234">
        <stream media="audio" dir="to-id1"/>
    </join>
    <join id1="conn:0010" id2=" conf:1234">
        <stream media="audio" dir="to-id1"/>
    </join>
    <join id1="conn:0011" id2=" conf:1234">
        <stream media="audio" dir="to-id1"/>
    </join>
    <join id1="conn:0012" id2=" conf:1234">
       <stream media="audio" dir="to-id1"/>
    </join>
</msml>
```

7.1.4 Call center coach-pupil conference

The following example uses the audio conference created in Creating a basic audio conference with <asn> and <n-loudest> for a call center application where a customer and an agent (pupil) are connected via the conference and a supervisor (coach) is also joined to the conference. The supervisor can hear both the customer and the agent. The supervisor can only be heard by the agent. The agent is heard by both the customer and the supervisor.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
    <join id1="conn:customer1" id2=" conf:1234"/>
    <join id1="conn:agent1" id2=" conf:1234">
       <stream media="audio" dir=”from-id1” dlgc:conf_party_type=”pupil”/>
       <stream media="audio" dir="to-id1"/>
    </join>
    <join id1="conn:supervisor1" id2=" conf:1234">
       <stream media="audio" dir=”from-id1” dlgc:conf_party_type=”coach”/>
       <stream media="audio" dir="to-id1"/>
    </join>
</msml>
```

7.1.5 Muting an audio stream flowing into a conference

This example mutes the audio stream from caller conn:0001 flowing into a conference.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
    <modifystream id1="conf:1234" id2=" conn:0001">
         <stream media=”audio” dir="to-id1">
              <gain amt="mute"/>
         </stream>
    </modifystream>
</msml>
```

This example mutes the audio stream from the conference flowing towards caller conn:0002.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
    <modifystream id1="conf:1234" id2=" conn:0002">
         <stream media=”audio” dir="from-id1">
             <gain amt="mute"/>
         </stream>
    </modifystream>
</msml>
```

7.1.6 Un-muting an audio stream flowing into a conference

A - This example un-mutes the audio stream from caller conn:0001 flowing into a conference using amt=”0”.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <modifystream id1="conf:1234" id2=" conn:0001">
         <stream media=”audio” dir="to-id1">
            <gain amt="0"/>
         </stream>
     </modifystream>
</msml>
```

This example un-mutes the audio stream from caller conn:0001 flowing into a conference using amt=”unmute”.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <modifystream id1="conf:1234" id2=" conn:0001">
         <stream media=”audio” dir="to-id1">
            <gain amt="unmute"/>
         </stream>
     </modifystream>
</msml>
```

B - This example un-mutes the audio stream from the conference flowing towards caller conn:0002 using amt=”0”.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <modifystream id1="conf:1234" id2=" conn:0002">
         <stream media=”audio” dir="from-id1">
            <gain amt="0"/>
         </stream>
     </modifystream>
</msml>
```

C - This example un-mutes the audio stream from the conference flowing towards caller conn:0002 using amt=”unmute”.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <modifystream id1="conf:1234" id2=" conn:0002">
        <stream media=”audio” dir="from-id1">
           <gain amt="unmute"/>
        </stream>
      </modifystream>
</msml>
```

7.1.7 Un-joining streams using wildcards

A - This example creates a full duplex audio stream between conf_1234 and conn:0001, creates a half duplex audio stream from conn:0002 to conn:0001, and creates a half duplex audio stream from conn:0002 to conf:1234.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
    <createconference name="1234" deletewhen="nomedia" term="false"/>
    <join id1="conn:0001" id2="conf:1234"/>
    <join id1="conn:0001" id2="conn:0002">
         <stream media="audio" dir="from-id1"/>
    </join>
    <join id1="conn:0002" id2="conf:1234">
         <stream media="audio" dir="from-id1"/>
    </join>
</msml>
```

B - After executing script A in this section, the following script un-joins all audio streams connected to conn:0001 (the full duplex connection between conn:0001 and conf:1234 and the half duplex stream from conn:0002 to conn:0001).

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <unjoin id1="conn:0001" id2="*"/>
</msml>
```

C - After executing script A in this section, the following script un-joins all audio streams connected to conf:1234 (the full duplex connection between conf:1234 and conn:0001 and the half duplex stream from conn:0002 to conf:1234).

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
      <unjoin id1="conf:1234" id2="*"/>
</msml>
```

D - After executing script A in this section, the following script un-joins all audio streams connected between conn:0001 and any conferencing object (the full duplex connection between conn:0001 and conf:1234).

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <unjoin id1="conn:0001" id2="conf:*"/>
</msml>
```

E - After executing script A in this section, the following script un-joins the audio stream from conf:1234 to conn:0001.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
    <unjoin id1="conf:1234" id2="*">
         <stream dir="from-id1"/>
    </unjoin>
</msml>
```

7.1.8 Continuous digit collection on a conference participant

This example illustrates when a call arrives at the media server, conn:0001. The following takes place:

1. A dialog (target conn:0001) is started that will collect digits one digit at a time and send an event to the application server for each digit received. This dialog executes continuously. It may be ended by the application at some point (not shown in the example).
2. A 2nd dialog (target conn:0001) is started that will play a prompt requesting the caller to enter the conference id. The prompt will be repeated up to three times.
3. The caller enters the conference id and these digits are sent to the application server.
4. The application, having received the conference id, then ends the 2nd dialog.
5. The application receives an event indicating that the play has ended and then receives the dialog.exit event for the 2nd dialog.
6. At this point, the application joins the caller (conn:0001) to the conference (conf:1234). The application will continue to receive dtmf digit events from the caller since the 1st dialog is still active.

   Note: Only 1 object or dialog can be transmitting to the network object conn:0001. If the application attempted to join the caller (conn:0001) to the conference before the 2nd dialog (which had been doing a play to the caller) had exited, the join operation would have resulted in an error.

Start continuous digit collection

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.0">
     <dialogstart target="conn:0001" type="application/moml+xml" name="dtmf.detect">
          <collect iterate="forever" cleardb="true">
               <pattern digits="x">
                   <send target="source" event="dtmf" namelist="dtmf.last"/>
               </pattern>
          </collect>
     </dialogstart>
</msml>
```

Play prompt requesting conference ID

```
<?xml version=”1.0” encoding=”UTF-8”?>
<msml version=”1.0”>
     <dialogstart target=”conn:0001” type=”application/moml+xml” name=”play.request_conf_id”>
          <play iterations="3" inteval="5s” barge=”false”>
              <audio uri=”http://host1 /conf_id_request.wav”/>
              <playexit>
                   <send target=”source” event=”playdone” namelist=”play.end”/>
              </playexit>
          </play>
      </dialogstart>
</msml>
```

Digit events are received (for conference id)
Digit event example

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <event name="dtmf" id="conn:0001/dialog:dtmf.detect'">
          <name>dtmf.last</name><value>1</value>
     </event>
</msml>
```

End dialog playing prompt requesting conference ID

```
<?xml version=”1.0” encoding=”UTF-8”?>
<msml version=”1.0”>
     <dialogend id=”conn:0001/dialog:play.request_conf_id”/>
</msml>
```

Playdone event is received

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <event name="playdone" id="conn:0001/dialog: play.request_conf_id">
          <name>play.end</name><value>play.terminate</value>
     </event>
</msml>
```

The dialog.exit event is received for the dialog playing prompt requesting conference ID

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <event name="msml.dialog.exit"
         id=" conn:0001/dialog: play.request_conf_id "/>
</msml>
```

Join conn:0001 to conf:1234

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <join id1="conn:0001" id2=" conf:1234”/>
</msml>
```

7.1.9 Destroying a conference

This example destroys conference conf:1234.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.0">
     <destroyconference id="conf:1234">
</msml>
```

7.2 MSML Scripts for Video Conferencing

The following sections provide examples of video conferencing tasks.

7.2.1 Creating a four party layout video conference

This example creates a video conference using a four party layout, as shown in the following figure, with a VGA root size, since 2 of the parties (conn:0001 and conn:0003) that have called into the video conference support H.264 VGA resolution. Caller conn:0002 uses H.264 QVGA and caller conn:0004 H.263 CIF:

Video Layout 4.1

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <createconference name="conf:876123" deletewhen="nomedia">
          <videolayout>
               <root size="VGA"/>
               <region id="1" left="0" top="0" relativesize="1/2"/>
               <region id="2" left="50%" top="0" relativesize="1/2"/>
               <region id="3" left="0" top="50%" relativesize="1/2"/>
               <region id="4" left="50%" top="50%" relativesize="1/2"/>
          </videolayout>
     </createconference>
</msml>
```

The four participants are then joined to the conference, regions 1, 2, 3 & 4:

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <join id1="conn:0001" id2="conf:876123">
         <stream media="audio"/>
         <stream media="video" dir="from-id1" display="1"/>
         <stream media="video" dir="to-id1"/>
     </join>
     <join id1="conn:0002" id2="conf:876123">
         <stream media="audio"/>
```

```
        <stream media="video" dir="from-id1" display="2"/>
        <stream media="video" dir="to-id1"/>
    </join>
    <join id1="conn:0003" id2="conf:876123">
        <stream media="audio"/>
        <stream media="video" dir="from-id1" display="3"/>
        <stream media="video" dir="to-id1"/>
    </join>
    <join id1="conn:0004" id2="conf:876123">
        <stream media="audio"/>
        <stream media="video" dir="from-id1" display="4"/>
        <stream media="video" dir="to-id1"/>
    </join>
</msml>
```

7.2.2 Expanding a four party layout to a six party layout video conference

This example modifies the video conference created in Creating a four party layout video conference from a four party layout to a six party layout while the four participants are still joined to the video conference.

Video Layout6.1

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
      <modifyconference id=" conf:876123">
           <videolayout>
               <root size="VGA"/>
               <region id="1" left="0" top="0" relativesize="2/3"/>
               <region id="2" left="66.666%" top="0" relativesize="1/3"/>
               <region id="3" left="66.666%" top="33.333%" relativesize="1/3"/>
               <region id="4" left="66.666%" top="66.666%" relativesize="1/3"/>
               <region id="5" left="33.333%" top="66.666%" relativesize="1/3"/>
               <region id="6" left="0" top="66.666%" relativesize="1/3"/>
           </videolayout>
      </modifyconference>
</msml>
```

Two additional participants are then joined to the conference, regions 5 & 6. Caller conn:0005 uses H.264 QVGA and caller conn:0006 H.263 CIF.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
      <join id1="conn:0005" id2="conf:876123">
           <stream media="audio"/>
           <stream media="video" dir="from-id1" display="5"/>
           <stream media="video" dir="to-id1"/>
      </join>
      <join id1="conn:0006" id2="conf:876123">
```

```
            <stream media="audio"/>
            <stream media="video" dir="from-id1" display="6"/>
            <stream media="video" dir="to-id1"/>
      </join>
</msml>
```

7.2.3 Contracting a six party layout to a four party layout video conference

This example modifies the video conference in Expanding a four party layout to a six party layout video conference from a six party layout to a four party layout after two callers, caller conn:0001 and caller conn:0003, leave the conference.

Caller conn:0001 and caller conn:0003, the only VGA callers, leave the conference:

Video Layout 4.1

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <unjoin id1="conn:0001" id2="conf:876123">
     </unjoin>
     <unjoin id1="conn:0003" id2="conf:876123">
     </unjoin>
</msml>
```

The conference is modified to a four party layout with a root size of QVGA. Regions 1 & 3 are made invisible (relativesize=”0”) and regions 2, 4, 5, and 6 are positioned as shown in the figure above.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <modifyconference id=" conf:876123">
          <videolayout>
              <root size="QVGA"/>
              <region id="1" left="0" top="0" relativesize="0"/>
              <region id="2" left="0" top="0" relativesize="1/2"/>
              <region id="3" left="66.666%" top="33.333%" relativesize="0"/>
              <region id="4" left="50%" top="0" relativesize="1/2"/>
              <region id="5" left="0" top="50%" relativesize="1/2"/>
              <region id="6" left="50%" top="50%" relativesize="1/2"/>
           </videolayout>
      </modifyconference>
</msml>
```

7.2.4 Layered regions in a video conference

This example modifies the four party video conference established in Creating a four party layout video conference to demonstrate use of the priority attribute to overlap regions.

First, region 1 is expanded to partially overlap the other 3 regions:

Video Layout 4.1

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <modifyconference name="conf:876123">
          <videolayout>
              <root size="VGA"/>
              <region id="1" left="0" top="0" relativesize="3/5" priority=”0.1”/>
              <region id="2" left="50%" top="0" relativesize="1/2"/>
              <region id="3" left="0" top="50%" relativesize="1/2"/>
              <region id="4" left="50%" top="50%" relativesize="1/2"/>
          </videolayout>
      </modifyconference>
</msml>
```

Second, region 1 is reduced to its original size and region 4 is expanded to partially overlap the other 3 regions:

Video Layout 4.1

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
      <modifyconference name="conf:876123">
           <videolayout>
                <root size="VGA"/>
                <region id="1" left="0" top="0" relativesize="1/2" priority=”1”/>
                <region id="2" left="50%" top="0" relativesize="1/2"/>
                <region id="3" left="0" top="50%" relativesize="1/2"/>
                <region id="4" left="60%" top="60%" relativesize="3/5" priority=”0.1”/>
           </videolayout>
       </modifyconference>
</msml>
```

7.2.5 Single party layout video conference using a <selector> element

The following examples create a video conference using a single party layout and a <selector> element that switches the party that is displayed based upon voice energy. Four parties are added to the conference.

Creating the conference

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <createconference name="conf:1111" deletewhen="nomedia">
           <videolayout>
               <selector id="switch1" method="vas">
                   <root size="CIF"/>
               </selector>
           </videolayout>
      </createconference>
</msml>
```

Joining four parties to the conference

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
     <join id1="conn:0001" id2=" conf:1111">
         <stream media="audio"/>
         <stream media="video" dir="from-id1" display="switch1"/>
         <stream media="video" dir="to-id1"/>
     </join>
     <join id1="conn:0002" id2=" conf:1111">
         <stream media="audio"/>
         <stream media="video" dir="from-id1" display=" switch1"/>
         <stream media="video" dir="to-id1"/>
     </join>
     <join id1="conn:0003" id2=" conf:1111">
         <stream media="audio"/>
         <stream media="video" dir="from-id1" display=" switch1"/>
         <stream media="video" dir="to-id1"/>
     </join>
     <join id1="conn:0004" id2=" conf:1111">
         <stream media="audio"/>
         <stream media="video" dir="from-id1" display=" switch1"/>
         <stream media="video" dir="to-id1"/>
     </join>
</msml>
```

7.2.6 Sequencing parties through regions in a video conference layout

For this example, a four-party layout is created as shown in Creating a four party layout video conference. A total of 16 parties will participate in the conference. Each region will be used to display four parties sequentially where each party will be visible for three seconds.

Parties are joined to the conference as they arrive as follows:

Join 16 parties (conn:0001 through conn:0016) as they arrive.
Parties: 1 (conn:0001), 5, 9, and 13 get joined to region 1.
Parties: 2 (conn:0002), 6, 10, and 14 get joined to region 2.
Parties: 3 (conn:0003), 7,11, and 15 get joined to region 3.
Parties: 4 (conn:0004), 8, 12, and 16 get joined to region 4.

Sending scripts with <unjoin> elements before doing the next <join> is not necessary. The last party joined to a region is the one that will be visible in that region.

Connect party 1 to region 1 as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
    <join id1="conn:0001" id2=" conf:876123">
         <stream media="audio" />
         <stream media="video" dir="from-id1" display="1"/>
         <stream media="video" dir="to-id1" />
    </join>
</msml>
```

To sequence through parties in a region, use <modifystream>. Sequence through parties 1, 5, 9, and 13 in region 1 displaying each party for approximately three seconds before displaying the next party:

Step A:

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
   <modifystream id1="conn:0001" id2=" conf:876123 ">
      <stream media="video" dir="from-id1" display="1"/>
      <stream media="video" dir="to-id1"/>
   </modifystream>
</msml>
```

Party 1 is now displayed in region 1. Wait three seconds and then proceed to Step B.

Step B:

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
   <modifystream id1="conn:0005" id2=" conf:876123">
      <stream media="video" dir="from-id1" display="1"/>
      <stream media="video" dir="to-id1"/>
   </modifystream>
</msml>
```

Party 5 is now displayed in region 1. Wait three seconds and then do Step C.

Step C:

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
    <modifystream id1="conn:0009" id2=" conf:876123">
       <stream media="video" dir="from-id1" display="1"/>
       <stream media="video" dir="to-id1"/>
    </modifystream>
</msml>
```

Party 9 is now displayed in region 1. Wait three seconds and then do Step D.

Step D:

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
    <modifystream id1="conn:0013" id2=" conf:876123">
        <stream media="video" dir="from-id1" display="1"/>
        <stream media="video" dir="to-id1"/>
    </modifystream>
</msml>
```

Party 13 is now displayed in region 1. Wait three seconds and then repeat Step A through Step D until all parties are sequenced.

7.2.7 Recording a segment of a video conference

```
<?xml version="1.0" encoding="utf-8" ?>
<msml version="1.0">
    <dialogstart target="conf:1234" type="application/moml+xml" name="DlgID">
        <record id="record1" maxtime="60s" audiodest="file:///root/ms/media/record1.pcm”
         videodest="file:///root/ms/media/record1.vid"
         format="video/proprietary;codecs=linear,h264" audiosamplerate="8"
         audiosamplesize="16"/>
    </dialogstart>
</msml>
```

7.2.8 Playing audio into a video conference

```
<?xml version="1.0" encoding="UTF-8" ?>
<msml version="1.1">
    <dialogstart target="conf:1234" type="application/moml+xml">
         <play barge="false" >
               <audio uri="file:///testing/media/conf_to_end.wav"/>
         </play>
    </dialogstart>
</msml>
```

Diagnostics 8

This chapter provides an overview of the diagnostic capabilities available in the MSML Media Server Software. Topics include:

- Overview 109
- Logging Configuration 109
- Log File Format 112

8.1 Overview

The media server is integrated with the RTF logging services used by other host libraries. The RTF XML configuration file, located in the *cfg* subdirectory, is used to set the logging level that defines what content is written to the RTF log file. The media server logging capabilities are described in the subsections following.

8.2 Logging Configuration

Different media server client objects can be assigned different logging levels. The following topics provide more detail:

- Configuration File Format
- Logging Labels
- Client Categories

8.2.1 Configuration File Format

The media server section of the RTF configuration file is given below. Detailed information about the logging labels and the clients that can be configured for logging is given in the sections following.

```
<!-- IP Media Server-->
<Module name="media_svr" state = "1">
   <MLabel  name="WARN" state = "0"/>
   <MLabel  name="INFO" state = "0"/>
   <MLabel  name="FUNC" state = "0"/>

        <!-- Media Server objects-->
   <MClient name="RESOURCE" state = "0"/>
   <MClient name="SIP_BOARD_RES" state = "0"/>
   <MClient name="SIP_RES" state = "0"/>
   <MClient name="IPM_RES" state = "0"/>
   <MClient name="DX_RES" state = "0"/>
   <MClient name="MM_RES" state = "0"/>
   <MClient name="CNF_RES" state = "0"/>
   <MClient name="CNF_PARTY_RES" state = "0"/>
```

```
    <MClient name="CNF_BOARD_RES" state = "0"/>
    <MClient name="FX_RES" state = "0"/>
    <MClient name="DT_RES" state = "0"/>
    <MClient name="MSML" state = "0"/>
    <MClient name="MOML" state = "0"/>
    <MClient name="NETWORK_CONN" state = "0"/>
    <MClient name="TDM_STREAM" state = "0"/>
    <MClient name="DEV_STREAM" state = "0"/>
    <MClient name="TRANSACTION" state = "0"/>
    <MClient name="MSML_TRANSACTION" state = "0"/>
    <MClient name="MOML_TRANSACTION" state = "0"/>
    <MClient name="XML_PARSER" state = "0"/>
    <MClient name="HTTP" state = "0"/>
    <MClient name="SOCKET" state = "0"/>
    <MClient name="EVENT_MGR" state = "0"/>
    <MClient name="TRANSACTION_MGR" state = "0"/>
    <MClient name="NETWORK_CONN_MGR" state = "0"/>
    <MClient name="SESSION_MGR" state = "0"/>
    <MClient name="MEDIA_STREAM_MGR" state = "0"/>
    <MClient name="RESOURCE_MGR" state = "0"/>
    <MClient name="THREAD_MGR" state = "0"/>
    <MClient name="CNF_MGR" state = "0"/>
    <MClient name="THREAD" state = "0"/>
    <MClient name="TIMER" state = "0"/>
    <MClient name="MEDIA_CODER" state="0" />
</Module>
```

8.2.2 Logging Labels

To enable media server logging in the RTF configuration file:

```
<Module name="media_svr" state = "1">
```

To disable all media server logging:

```
<Module name="media_svr" state = "0">
```

Note: Special startup/shutdown configuration information is always logged.

The logging labels that can be specified for the various media server client objects are as follows:

ERROR - Default
: Log unexpected conditions or events that cause a related action to fail. This is a global label which is set in the beginning of the RTF XML file.

WARN
: Log unexpected conditions or events that should not cause a related action to fail.

INFO
: Log additional information such as state changes, events, and function parameters.

FUNC
: Logs function entry and exit.

Note: Logging labels are independent of each other. For example, enabling FUNC and disabling INFO logs function entry and exit, but not additional information.

8.2.3 Client Categories

The various clients that can be enabled are listed below. Each client can be set to 0 (disable) or 1 (enable). If enabled, the logging labels determine the detail of the log content. The clients can be categorized as follows:

SIP Call Control

Provides the call control implementation that allows call establishment with application servers:

- SIP_BOARD_RES – ipt virtual board abstraction
- SIP_RES – ipt channel
- NETWORK_CONN – network connection object that uses a SIP resource and conditionally an ipm resource to establish a call; also responsible for processing SDP information
- NETWORK_CONN_MGR – manager that creates, initializes and destroys network connections
- OFFER_ANSWER – Offer Answer object that handles coder negotiation

Resources

Abstractions of the media resources provided by Dialogic:

- RESOURCE – base object from which all other resources are derived
- IPM_RES – ipm resource
- DX_RES – dxx resource
- MM_RES – mm resource
- CNF_RES – cnf resource
- CNF_BOARD_RES – cnf virtual board resource
- CNF_PARTY_RES – cnf party resource
- CNF_MNR – cnf manager, responsible for mapping conference names to resources and providing bridge support
- FX_RES – fax object; currently not supported
- RESOURCE_MGR – creates, allocates, deallocates and destroys all resources

MSML MOML

Objects responsible for processing MSML specific requests:

- MSML – MSML object
- MOML – MOML object

Connections

Objects responsible for creating, modifying, and destroying resource connections including gain and AGC:

- TDM_MEDIA_STREAM – implements TDM listen/unlisten functionality
- DEV_MEDIA_STREAM – implements dev_Connect/dev_Disconnect functionality
- MEDIA_STREAM_MGR – manages DEV and TDM streams
- PORT_STREAM – implements dev_PortConnect/dev_PortDisconnect functionality

Transactions

Requests sent to the media server to perform media operations:

- TRANSACTION – base object from which all other transactions are derived
- MSML_TRANSACTION – MSML transactions
- MOML_TRANSACTION – not currently used
- TRANSACTION_MGR – allocates and deallocates transactions

Utilities

Objects that provide internal services required by clients:

- TIMER – provides timer callback trigger
- THREAD – provides a separate thread context to execute some action
- THREAD_MGR – allocates/deallocates threads
- HTTP – read from or write to remote files
- SOCKET – read from or write to TCP/IP socket; currently not supported

XML Parsing

Objects for XML parsing:

- XML_PARSER – XML parser

Miscellaneous

Other objects:

- EVENT_MGR – Standard Runtime Library (SRL) event manager; dispatches SRL events to registered clients
- SESSION_MGR – creates and destroys all other managers

8.3 Log File Format

The following is a snippet from a sample log file.

```
. . .

06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !               MSML !    1 ! ====> CMSML::ValidateScript()
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !               MSML !               MSML !    1 ! CMSML::ValidateScript - Dumping
script body
                                                 <?xml version="1.0" encoding="UTF-8" ?>
                                                 <msml version="1.1">
                                                 <join id1="conn:51f0188-0-13d8-64325-7d02fa29-64325" id2="conn:51f0698-0-13d8-
64334-29ea58e0-64334" />
                                                 </msml>
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !               MSML !    1 ! <==== CMSML::ValidateScript()
returns 0
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !               MSML !               MSML !    1 ! CMSML::EvProcessScript - Validation
successful.  Processing initiated
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !               MSML !               MSML !    1 ! CMSML::EvProcessScript - Mandatory
attribute version = 1.0
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !               MSML !    1 ! ====> CMSML::ProcessElement()
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !               MSML !               MSML !    1 ! CMSML::ProcessElement - Processing
new MSML element. Element = join
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !               MSML !    1 ! ====> CMSML::ProcessEvent(), event =
1
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !               MSML !               MSML !    1 ! CMSML::ProcessEvent - Processing
Event EV_Join in State MSML_Processing
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !               MSML !    1 ! ====> CMSML::SetState()
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !               MSML !               MSML !    1 ! CMSML::SetState - Transitioning from
MSML_Processing to MSML_Connecting
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !               MSML !    1 ! <==== CMSML::SetState()
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !               MSML !    1 ! ====> CMSML::EvJoin()
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !           MSMLJoin !    1 ! ====> CMSMLJoin::Parse()
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !           MSMLJoin !    1 ! ====> CMSMLJoin::Reset()
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !           MSMLJoin !    1 ! <==== CMSMLJoin::Reset() returns 0
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !               MSML !           MSMLJoin !    1 ! CMSMLJoin::Parse - Processing MSML
element <join>
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !           MSMLJoin !    1 ! ====>
CMxMLElement::LoadMandatoryAttribute()
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !               MSML !           MSMLJoin !    1 ! CMxMLElement::LoadMandatoryAttribute
- Mandatory attribute id1 = conn:51f0188-0-13d8-64325-7d02fa29-64325
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !           MSMLJoin !    1 ! <====
CMxMLElement::LoadMandatoryAttribute()
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !           MSMLJoin !    1 ! ====>
CMxMLElement::LoadMandatoryAttribute()
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !               MSML !           MSMLJoin !    1 ! CMxMLElement::LoadMandatoryAttribute
- Mandatory attribute id2 = conn:51f0698-0-13d8-64334-29ea58e0-64334
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !           MSMLJoin !    1 ! <====
CMxMLElement::LoadMandatoryAttribute()
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !               MSML !           MSMLJoin !    1 ! ====>
CMxMLElement::LoadOptionalAttribute()
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !               MSML !           MSMLJoin !    1 ! CMxMLElement::LoadOptionalAttribute
```

```
- Optional attribute mark = null
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !              MSML !          MSMLJoin !    1 ! <====
CMxMLElement::LoadOptionalAttribute()
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !              MSML !          MSMLJoin !    1 ! <==== CMSMLJoin::Parse() returns 0
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !              MSML !          MSMLJoin !    1 ! ====> CMSMLJoin::Execute()
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !              MSML !          MSMLJoin !    1 ! ====> CMSMLJoin::CreateStreams()

06/19/2006 09:12:23.035   1300   740 media_svr    INFO !              MSML !          MSMLJoin !    1 ! CMSMLJoin::CreateStreams -
Initiating connection between conn:51f0188-0-13d8-64325-7d02fa29-64325 and conn:51f0698-0-13d8-64334-29ea58e0-64334 via MediaStreamMgr
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! ===> CreateStream()
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! Create 2 streams,
IClientNotify=0x5356284
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! ===> SortStreams()
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! Sort 2 SMediaStreams
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! <=== SortStreams(): return E_SUCCESS
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! ===> AreStreamsActive()
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! Check if 2 TDM streams and 0 DEV
streams are active
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! <=== AreStreamsActive(): returns
false
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! ===> AllocateCxTransaction()
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! <==== AllocateCxTransaction()
returns 0
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! ===> AllocateCTDMStream()
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! AllocateCTDMStream() return
CMediaStream with Id = 20
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! <==== AllocateCTDMStream() returns 0
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! Create TDM Stream, TxResource =
ipmB1C49,  RxResource = ipmB1C47
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! ===> AllocateCTDMStream()
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! AllocateCTDMStream() return
CMediaStream with Id = 19
06/19/2006 09:12:23.035   1300   740 media_svr    FUNC !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! <==== AllocateCTDMStream() returns 0
06/19/2006 09:12:23.035   1300   740 media_svr    INFO !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! Create TDM Stream, TxResource =
ipmB1C47,  RxResource = ipmB1C49
06/19/2006 09:12:23.115   1300   2260 media_svr   FUNC !       NETWORK_CONN ! CNetworkConnection !    3 ! ====> Notify()
06/19/2006 09:12:23.115   1300   2260 media_svr   INFO !       NETWORK_CONN ! CNetworkConnection !    3 ! Notify - IPMEV_LISTEN Event  -
Device = ipmB1C47
06/19/2006 09:12:23.115   1300   2260 media_svr   FUNC !       NETWORK_CONN ! CNetworkConnection !    3 ! <==== Notify() returns 8.
06/19/2006 09:12:23.115   1300   2260 media_svr   FUNC !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! ====> Notify()
06/19/2006 09:12:23.115   1300   740 media_svr    FUNC !   MEDIA_STREAM_MGR !    CMediaStreamMgr !    1 ! <=== CreateStream(): returns
E_SUCCESS
. . .
```

Media Server Markup Language (MSML) Overview A

This chapter provides a brief overview of the Media Server Markup Language (MSML). The descriptions are based on *Media Server Markup Language (MSML)*, Internet IETF RFC5707 and are considered the nominal. Topics include:

- Introduction 115
- MSML Elements 115
- Stream Manipulation Elements 116
- Conference Elements 117
- Dialog Elements 117
- Receiving Events from a Client 117
- Sending Events and Transaction Results to a Client 118
- Media Server Object Model 118

A.1 Introduction

The Media Server Markup Language (MSML) is an XML [n2] language used to control the flow of media streams and services applied to media streams within a media server. It provides a means to configure, define and control media resource objects to construct many different types of services on individual sessions, groups of sessions, and conferences. MSML allows the creation of conferences, bridging different sessions together, and bridging sessions into conferences. Additionally MSML facilitates the use of complex interactions between media objects and constructs to create and control media processing operations used for application interactions with end users (e.g., announcements, Interactive voice response (IVR, IVVR), play and record, automatic speech recognition (ASR), text to speech (TTS)).

Readers who want an in-depth understanding of the control language should refer to IETF standard RFC 5707.

Note: The current implementation of the MSML Media Server Software does not support all the capabilities of MSML.

A.2 MSML Elements

MSML commands are sent from a client to the MS via SIP messages (most notably the INFO message). The body of the SIP message contains the XML control syntax.

A.2.1 <msml>

The root XML element of MSML is <**msml**>. It defines the set of operations that form a single MSML transaction.

Results or events returned to the client are also enclosed in the <**msml**> element.

A.2.2 <send>

The <**send**> element is used by a client to send an event to the MS.

A.2.3 <result>

The <**result**> element is used by the MS to report the results of an MSML transaction requested by a client.

A.2.4 <event>

The <**event**> element is used by the MS to notify a client of an event.

A.3 Stream Manipulation Elements

The following subsections describe the elements that establish, modify, and remove streams.

Note: The <**monitor**> element described in MSML IETF draft, version -06 is **not** currently supported.

A.3.1 <join>

The <**join**> element is used to create one or more streams between two independent objects. Streams may be audio or video and may be unidirectional or bidirectional.

A.3.2 <modifystream>

The <**modifystream**> element is used to change the properties of a stream. The <**modifystream**> element can have different properties such as the gain for an audio stream or a visual label for a video stream.

A.3.3 <unjoin>

The <**unjoin**> element is used to remove one or more existing media streams between two objects.

A.4 Conference Elements

The following subsections describe the elements that establish, modify, and destroy conferences.

A.4.1 <createconference>

The <**createconference**> element is used to create a conference. The MS assigns a conference name if the name attribute is not included.

Note: Only audio conferences are currently supported.

A.4.2 <modifyconference>

The <**modifyconference**> element is used to change the properties of a conference, such as the active talker interval.

A.4.3 <destroyconference>

The <**destroyconference**> element is used to delete an existing conference.

A.5 Dialog Elements

Dialogs are used for interaction with a user.

A.5.1 <dialogstart >

The <**dialogstart**> element is used to instantiate a media dialog on connections or conferences.

A.5.2 <dialogend >

The <**dialogend**> element is used to terminate a dialog created through <**dialogstart**>.

A.6 Receiving Events from a Client

Events are received from clients via SIP INFO messages. Events are used to affect the behavior of different objects within the MS. The client includes the <**send**> element within the <**msml**> root element. The <**send**> element identifies the event to process.

A.7 Sending Events and Transaction Results to a Client

A.7.1 Transaction Results

The **<result>** element is used to report the results of an MSML transaction. The **<result>** element is included in the final response to the SIP INFO message that initiated the transaction.

A.7.2 Events

The **<event>** element is used to notify the MS client of an event. Events are sent to clients via SIP INFO messages.

A.8 Media Server Object Model

Media server objects represent entities that source, sink, or modify media streams. A media stream is a unidirectional or bidirectional media flow between objects in a MS.

The MS object classes are:

- Network Connections (conn)
- Conference (conf)
- Dialog (dialog)
- Operator (oper)

A.8.1 Network Connections (conn)

Definition

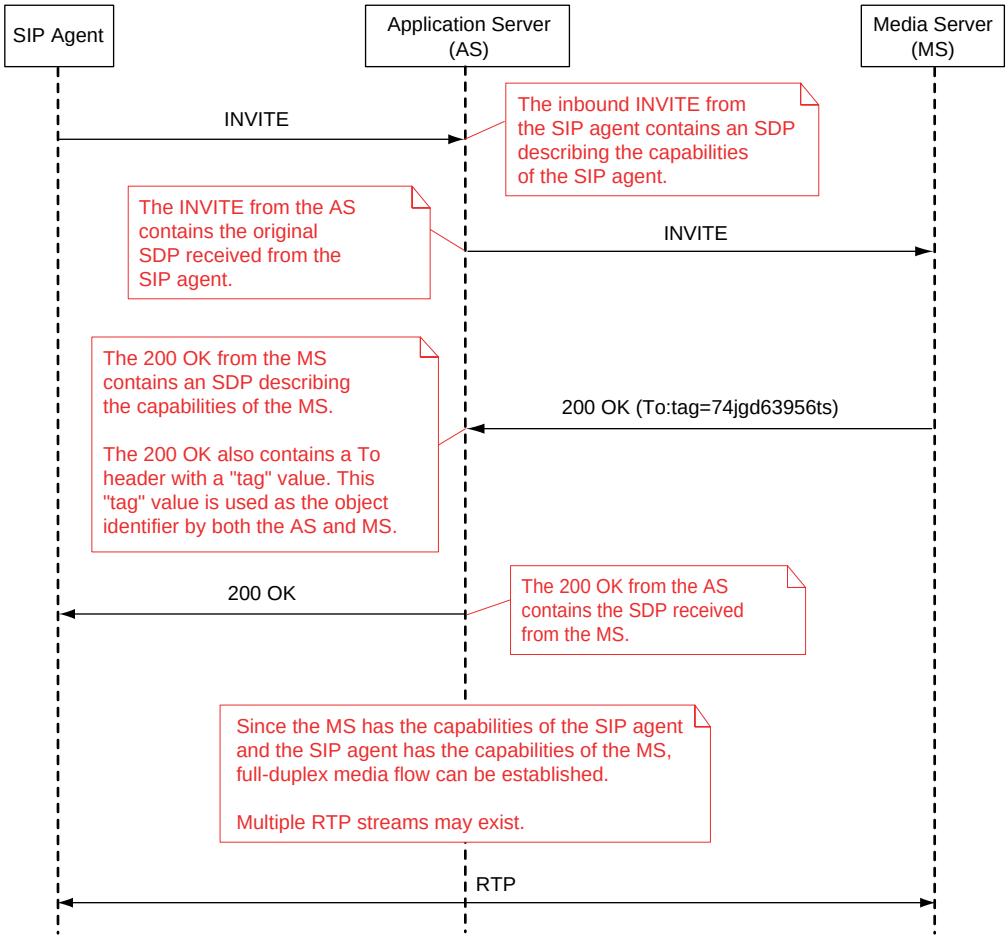
A Network Connection is an object or class defined in the MSML specification. Network Connection is an abstraction for the media processing resources involved in terminating the RTP session(s) of a call. For audio services, a connection instance presents a full-duplex audio stream interface within a MS. Multimedia connection objects have multiple media streams of different media types, each corresponding to an RTP session. MSML Network Connection instances are instantiated through SIP.

Object Creation

Unlike other MSML objects that are created using MSML commands/elements, Network Connection objects are not created using MSML commands/elements. Network Connections are created when media sessions get established through SIP call control. The connection identifier is not assigned by the AS. It is assigned by the MS and is returned to the AS in the response to the initial INVITE received from the AS. Specifically, this is the “tag” value included in the “To” header in the response. The format of the connection identifier is “conn:<tag value>”.

Figure 3 shows the interactions between the MS and the AS to create a Network Connection and establish an object identifier.

Figure 3. Network Connection Creation



Note: In Figure 3, the identifier used by the MS and AS to reference the network connection is "conn:74jgd63956ts".

A.8.2 Conference (conf)

Definition

A Conference is an object or class defined in the MSML specification that allows for audio/video mixing and other advanced conferencing services. A conference represents the media resources and state information required for a single logical mix of each media type in the conference (for example, audio and video). A conference has a single logical output per media type. For each participant, it consists of the audio conference mix, less any contributed audio of the participant, and the video mix shared by all conference participants.

Object Creation

Conferences are created using the **<createconference>** MSML command. The conference name can be assigned by the AS or, if the AS does not specify a name, the MS assigns one. Both the AS and the MS reference to the conference using the name as the ID, conf="name".

The AS can request that a MS automatically delete a conference when a specified condition occurs by using the **deletewhen** attribute. A value of "nomedia" indicates that the conference must be deleted when there are no remaining participants in the conference. When this occurs, an "msml.conf.nomedia" event must be notified to the MSML client. A value of "nocontrol" indicates that the conference must be deleted when the SIP dialog that carries the **<createconference>** element is terminated. When this occurs, a MS must terminate all participant dialogs by sending a BYE for their associated SIP dialog. A value of "never" must leave the ability to delete a conference under the control of the MSML client.

Additional content of the **<createconference>** element specifies the parameters of the audio and/or video mixes.

An example of the creation of an audio conference is shown below. This conference reports the set of active speakers no more frequently than every 10 seconds.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
    <createconference name="example">
        <audiomix>
            <asn ri="10s"/>
        </audiomix>
    </createconference>
</msml>
```

A.8.3 Dialog (dialog)

Definition

Dialogs are a class of objects that represent automated participants. Dialogs are similar to network connections from a media flow perspective and may have one or more media streams as the abstraction for their interface within the MS. Unlike network connections, dialogs are created and destroyed through MSML. The MS implements the dialog participant.

A Dialog is a generic reference to the set of resources, both media and control, that are used to create a simple or complex action. An atomic play or record is an example of a simple action.

The function that a Dialog instance fulfills is defined by a client and the language utilized. In this case, it is MOML.

MSML Dialog instances are instantiated through the **<dialogstart>** element.

Object Creation

All MSML objects except the Network Connection objects are created using MSML commands/elements.

The following example starts a MOML dialog on a connection.

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
   <dialogstart target="conn:abcd1234"
      type="application/moml+xml"
      name="sample"
      src="http://server.example.com/scripts/foo.moml"/>
</msml>
```

A.8.4 Operator (oper)

Definition

An Operator is an object or class used to filter or transform a media stream. Operators have a media type and may be unidirectional or bidirectional.

Unidirectional operators reflect simple atomic functions, such as automatic gain control or filtering tones. Unidirectional operators have a single media input that is connected to the media stream from one object, and a single media output that is connected to the media stream of a different object.

Bidirectional operators have two media inputs and two media outputs. One media input and output is associated with the stream to one object, and the other input and output is associated with a stream to a different object.

The function that an Operator instance fulfills is defined by a client and the language utilized. In this case, it is MOML.

MSML Operator instances are instantiated when streams are created using a <**join**> element or modified using a <**modifystream**> element.

Object Creation

All MSML objects except Network Connection objects are created using MSML commands/elements.

Glossary

3PCC: Third-Party Call Control

Application Server: An external server running applications that originate MSML requests to a Media Server (MS).

IETF: Internet Engineering Task Force

IVR: Interactive Voice Response

IP: Internet Protocol

Media Server: A general-purpose platform for executing real-time media processing tasks. It may be a single physical device or a logical function within a physical device.

MOML: Media Objects Markup Language

MSML: Media Sessions Markup Language

RTP: Real Time Protocol

SDP: Session Description Protocol

SIP: Session Initiation Protocol

XML: Extensible Markup Language