Dialogic®

Dialogic® PowerMedia™ XMS MSML Media Server Software

User's Guide

July 2016 05-2717-008

www.dialogic.com

Copyright and Legal Notice

Copyright © 2008-2016 Dialogic Corporation. All Rights Reserved. You may not reproduce this document in whole or in part without permission in writing from Dialogic Corporation at the address provided below.

All contents of this document are furnished for informational use only and are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation and its affiliates or subsidiaries ("Dialogic"). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Dialogic does not warrant the accuracy of this information and cannot accept responsibility for errors, inaccuracies or omissions that may be contained in this document.

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

Dialogic products are not intended for use in certain safety-affecting situations. Please see http://www.dialogic.com/company/terms-of-use.aspx for more details.

Due to differing national regulations and approval requirements, certain Dialogic products may be suitable for use only in specific countries, and thus may not function properly in other countries. You are responsible for ensuring that your use of such products occurs only in the countries where such use is suitable. For information on specific products, contact Dialogic Corporation at the address indicated below or on the web at www.dialogic.com.

It is possible that the use or implementation of any one of the concepts, applications, or ideas described in this document, in marketing collateral produced by or on web pages maintained by Dialogic may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic does not provide any intellectual property licenses with the sale of Dialogic products other than a license to use such product in accordance with intellectual property owned or validly licensed by Dialogic and no such licenses are provided except pursuant to a signed agreement with Dialogic. More detailed information about such intellectual property is available from Dialogic's legal department at 6700 Cote-de-Liesse Road, Suite 100, Borough of Saint-Laurent, Montreal, Quebec, Canada H4T 2B5. **Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.**

Dialogic, Dialogic Pro, Dialogic Blue, Veraz, Brooktrout, Diva, BorderNet, PowerMedia, PowerVille, PowerNova, MSaaS, ControlSwitch, I-Gate, Mobile Experience Matters, Network Fuel, Video is the New Voice, Making Innovation Thrive, Diastar, Cantata, TruFax, SwitchKit, Eiconcard, NMS Communications, SIPcontrol, Exnet, EXS, Vision, inCloud9, NaturalAccess and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic Corporation and its affiliates or subsidiaries. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic's legal department at 6700 Cote-de-Liesse Road, Suite 100, Borough of Saint-Laurent, Montreal, Quebec, Canada H4T 2B5. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

The names of actual companies and products mentioned herein are the trademarks of their respective owners.

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

Table of Contents

Revision History

Revision	Release Date	Notes
05-217-008 (Updated)	July 2016	MSML Dialog Core Package Support: Added details regarding the target attribute for the <send> element. Differences between Native and Legacy MSML: Updated the section. Pattern Matching and Digit Buffer Rules: • Added details regarding the digits attribute of the <pattern> element. • Added details regarding the edt timer. Call Progress Analysis (CPA) Support: Added the section.
05-217-008 (Updated)	May 2016	Feature Details: Added the MSML Schema Validation section and removed the related content from the Appendix.
05-217-008 (Updated)	April 2016	Removed WebRTC support.
05-217-008 (Updated)	August 2015	Feature Details: • Updated the MinMax information in the Pattern Matching and Digit Buffer Rules section.
05-2717-008 (Updated)	July 2015	Feature and Protocol Package Support: • Updated the prespeech attribute in the Dialog Base Package Support table. Deviations: • Added "no-ringback" value to cpa.detect. Feature Details: • Updated the Media Stream Direction Support section. MSML Script Examples: • Updated the Call center coach-pupil conference script. • Updated the Setting <stream> attribute for echo cancellation script. Media Server Markup Language (MSML) Overview: • Added XML Schema information.

Revision	Release Date	Notes
05-2717-008 (Updated)	June 2015	Feature and Protocol Package Support: • Updated the MSML Dialog Base Package Support table to add a comment that append attribute in the <record> element supports audio only. Feature Details: • Added convert option to the operation attribute for <fileop> element in Session File Transfer section. • Added srcformat and destformat attributes for <fileop> element in Session File Transfer section.
05-2717-008 (Updated)	May 2015	Feature and Protocol Package Support: • Added the MSML Dialog Speech Package Support and MSML Dialog Fax Detection Package Support tables. Deviations: • Updated section for Differences between Native and Legacy MSML. MSML Script Examples: • Updated Destroying a conference example in MSML Scripts for Audio Conferencing section. Media Server Markup Language (MSML) Overview: • Updated the Video Play - Dialogic VID (proprietary) table in Media File Formats section to correct MSML attribute format for jpeg codec.
05-2717-008	March 2015	Updates to support PowerMedia XMS Release 2.4. Feature and Protocol Package Support: • Updated the MSML Dialog Base Package Support table to change record.recordid shadow variable in the <record> element as not supported. Feature Details: • Updated section for Session File Transfer. • Updated section for Media Stream Direction Support with details about WebRTC. • Added section for DTMF Clamping for <record>. • Added section for Multiple URI for <audio>

Revision	Release Date	Notes
		and <video>. • Added section for 3GP Multimedia Container. • Added section for Simultaneous Dual file (A+A/V) 3GP Record Mode. MSML Script Examples: • Added Voice activated switching examples in MSML Scripts for Video Conferencing section. Media Server Markup Language (MSML) Overview: • Updated the various tables in Media File Formats section with 3GP container.
05-2717-007	January 2015	Media Server Markup Language (MSML) Overview: • Added dlgc:target_display attribute to <dialogstart> element in Dialog Elements section. • Updated the Video Record table in Media File Formats section.
05-2717-006	December 2014	Configuration: • Updated the details for configuring MSML. Feature and Protocol Package Support: • Updated the various tables. • Updated the tables with support for id attribute of the <gain> element. Deviations: • Updated section for Differences between Native and Legacy MSML. Feature Details: • Updated section for Session File Transfer. • Updated section for Pattern Matching with <dtmf>/<collect>. • Added section for Monitoring RTP Timeout Alarms.
05-2717-005	October 2014	Updates to support PowerMedia XMS Release 2.3. Configuration: • Updated valid values for Media Mode Selection parameter in MSML Configuration. • Added table for Media Mode Combinations. • Added section for Alarms in MSML Advanced

Revision	Release Date	Notes
		Configuration. Feature and Protocol Package Support: • Updated the MSML Dialog Core Package Support table with support for <exit> element and namelist attribute. • Updated the MSML Dialog Core Package Support table with support for <disconnect> element and namelist attribute. • Updated the MSML Dialog Base Package Support table with support for terminate.cancelled and terminate.finalsilence events in the <record> element. • Updated the MSML Dialog Base Package Support table with support for edt event in the <dtmf> and <collect> elements. Deviations: • Added items in the Deviations from RFC 5707 section. • Added items in the Differences between Native and Legacy MSML section. Feature Details: • Added section for Session File Transfer with support for <transfer> and <fileop> elements. • Added section for Pattern Matching with <dtmf>/<collect>. Sample Use Case: • Added note about I-frame to Step 5g for Record Message. MSML Script Examples: • Updated Start continuous digit collection example in Continuous digit collection on a conference participant. • Added section for MSML Scripts Examples for <transfer> element. Media Server Markup Language (MSML) Overview: • Added details about codec feature in Media File Formats section. • Updated the Media File Formats table with AMR and AMR-WB containers. • Added the Audio Play - AMR, Audio Play - AMR-WB, Audio Record - AMR, and Audio Record - AMR-WB tables in Media File Formats

Revision	Release Date	Notes
		section.
05-2717-004	May 2014	Feature and Protocol Package Support: • Updated the size attribute with additional supported values in <root> section. Deviations: • Added item in the Differences between Native and Legacy MSML section.
05-2717-003	March 2014	Updates to support PowerMedia XMS Release 2.2. Configuration: • Added note for the Schema Validation parameter in MSML Configuration Parameters table. Deviations: • Added item in the Differences between Native and Legacy MSML section. Media Server Markup Language (MSML) Overview: • Updated the tables in Media File Formats section.
05-2717-002	November 2013	Deviations: • Added section for Differences between Native and Legacy MSML. Media Server Markup Language (MSML) Overview: • Added section for Media File Formats.
05-2717-001	October 2013	Updates to support PowerMedia XMS Release 2.1.
05-2717-001-01	August 2013	Global change: • Renamed this document from Dialogic® MSML Media Server Software User's Guide to Dialogic® PowerMedia™ XMS MSML Media Server Software User's Guide.
Last modified: July 2016		

Refer to www.dialogic.com for product updates and for information about support policies, warranty information, and service offerings.

Welcome

This User's Guide provides information about the Dialogic® PowerMedia™ Extended Media Server (also referred to herein as "PowerMedia XMS" or "XMS") Media Sessions Markup Language (MSML) software.

The MSML Media Server software enables a remote client, also known as an application server (AS), to control media resources on a media server (MS). 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.

About This Publication

The following topics provide information about this publication.

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 Media Server (MS).

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™ Extended Media Server (also referred to herein as "PowerMedia XMS" or "XMS").

Additionally, this publication documents Dialogic's compliance with the RFC 5707 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 provided in a phased approach. A phase typically introduces support for a previously unsupported package(s), element(s) or attribute(s). This publication documents the functionality provided by the current set of supported MSML packages as described in RFC 5707 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 Base Package
- MSML Dialog Core Package
- MSML Dialog Fax Detection Package
- MSML Dialog Group Module Package (parallel topology only)
- MSML Dialog Speech Package
- MSML Dialog Transform Primitives Module Package (gain only)

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

- 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 RFC 3261.

How to Use This Publication

This publication is divided into the following sections:

- MSML Media Server Software Overview describes the role of the MSML Media Server Software in a Media Server environment.
- Configuration explains how to configure the MSML Media Server Software for operation on a Media Server.
- Feature and Protocol Package Support specifies high-level feature support by platform and identifies packages, elements, and attributes (as documented in the RFC 5707 MSML specification) currently supported or not supported by the MSML Media Server Software.
- Deviations explains deviations from the RFC 5707 MSML specification.
- Feature Details provides details on features supported, including a feature description and how-to information.
- Sample Use Case presents an application that demonstrates many of the features currently supported by the MSML Media Server Software.
- Diagnostics describes the logging capabilities available to the MSML Media Server Software for diagnostic purposes.
- 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)

1. MSML Media Server Software Overview

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

Introduction

The MSML Media Server Software is an integral part of the system software provided by PowerMedia XMS.

When the PowerMedia XMS 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 Media Server Markup Language (MSML) as defined in the RFC 5707 MSML specification, 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.

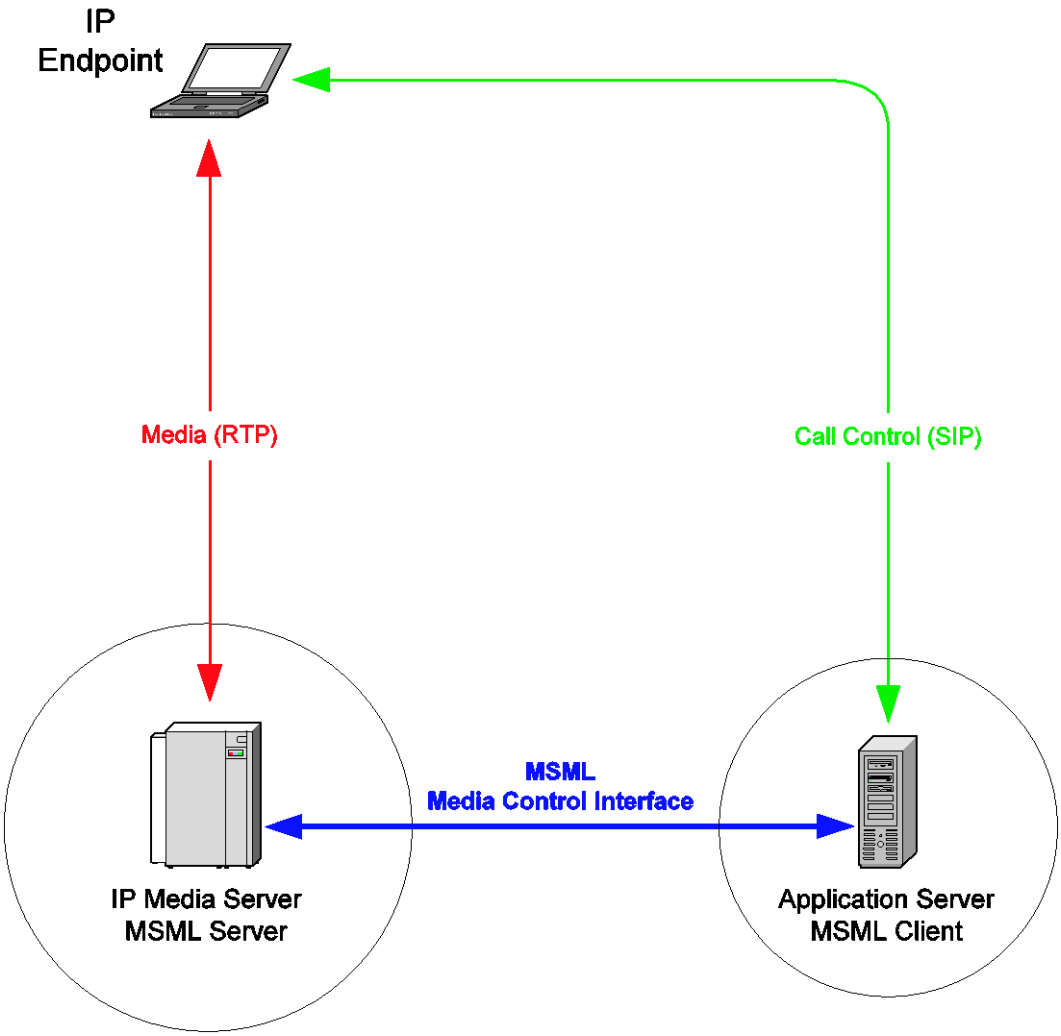
Media Server Operating Model

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 an 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.

The MSML interface uses SIP INFO messages to send MSML script payloads.

Figure 1. Media Server Operating Environment



2. Configuration

This chapter discusses configuration topics, such as how to configure PowerMedia XMS to use the MSML Media Server Software.

- Configuring PowerMedia XMS
- Configuring MSML

Configuring PowerMedia XMS

MSML installation can be verified with moml=file:///var/lib/xms/msml/verification.moml parameter in the SIP INVITE Request-URI. Playing the verification prompt is an indication of a successful installation.

PowerMedia XMS configuration and operation is done though a secure web-based operator console called the PowerMedia XMS Admin Console (also referred to herein as "Console").

The Console can be reached using a web browser and the PowerMedia XMS IP address. For more information on how to access and use the Console, see the *Dialogic® PowerMedia™ XMS Installation and Configuration Guide*.

Configuring MSML

The MSML interface (RFC 5707) uses SIP INFO messages to send MSML script payloads.

The **MSML** menu contains tabbed pages, **MSML Configuration** and **MSML Advanced Configuration**.

For more information on how to configure MSML, see the *Dialogic® PowerMedia™ XMS Installation and Configuration Guide*.

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

The MSML Media Server Software level of support varies in conjunction with the associated PowerMedia XMS 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	PowerMedia XMS	Feature Details/Comments
Audio conferencing	Supported	
Audio play and record	Supported	
Audit package	Supported	
Digit detection	Supported	
Digit detection - RFC 2833	Supported	
HTTP play and record	Supported	See HTTPS Play and Record.
Text to speech and speech recognition	Supported	
Fax detection	Supported	
File transfer	Supported	See Session File Transfer.
Video play and record	Supported	
Video conferencing	Supported	

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 RFC 5707. 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 MSML Core Package Support.
Section 8	MSML Conference Core Package	Conditionally Mandatory, for Conferencing	Supported See MSML Conference Core Package Support.
Section 9.6	MSML Dialog Core Package	Conditionally Mandatory, for Dialogs	Supported See MSML Dialog Core Package Support.
Section 9.7	MSML Dialog Base Package	Conditionally Mandatory, for Dialogs	Supported See MSML Dialog Base Package Support.
Section 9.8	MSML Dialog Group Package	Optional	Supported See MSML Dialog Group Package Support.
Section 9.9	MSML Dialog Transform Package	Optional	Supported See MSML Dialog Transform Package Support.
Section 9.10	MSML Dialog Speech Package	Optional	Supported See MSML Dialog Speech Package Support.

RFC 5707 Ref	Package Name	Requirement	Level of Support
Section 9.11	MSML Dialog Fax Detection Package	Optional	Supported See MSML Dialog Fax Detection Package Support.
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 MSML Audit Core Package Support.
Section 10.2	MSML Audit Conference Package	Conditionally Mandatory, for Auditing Conference, Conference Dialog, and Conference Stream	Supported See MSML Audit Conference Package Support.
Section 10.3	MSML Audit Connection Package	Conditionally Mandatory, for Auditing Connection, Connection Dialog, and Connection Stream	Supported See MSML Audit Connection Package Support.
Section 10.4	MSML Audit Dialog Package	Conditionally Mandatory, for Auditing Dialog	Supported See MSML Audit Dialog Package Support.
Section 10.5	MSML Audit Stream Package	Conditionally Mandatory, for Auditing Stream	Supported See MSML Audit Stream Package Support.

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	

RFC 5707 Ref	Element Name	Attribute Name	Level of Support	Comments
		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	

RFC 5707 Ref	Element Name	Attribute Name	Level of Support	Comments
Section 8.4	<modifyconference>	-	supported	
		id	supported	
		mark	supported	
Section 8.5	<destroyconference>	-	supported	
		id	supported	
		mark	supported	
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	
Section 8.7.2	<region>	-	supported	

RFC 5707 Ref	Element Name	Attribute Name	Level of Support	Comments
		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	
		id	supported	
		method	supported	Supports "vas" only.
		status	not supported	
		blankothers	not supported	

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	

RFC 5707 Ref	Element Name	Attribute Name	Level of Support	Comments
		media	supported	
		dir	supported	
		compressed	supported	
Section 8.12.1 Audio Stream Properties		preferred	supported	
		dlgc:conf_party_type	supported	This is a Dialogic extension. See <stream>.
		dlgc:echo_cancel	supported	This is a Dialogic extension. See <stream>.
Section 8.12.1.1	<gain>	-	supported	
		id	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 Video Stream Properties		display	supported	
		override	not supported	

RFC 5707 Ref	Element Name	Attribute Name	Level of Support	Comments
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	
		dlgc:target_display	supported	This is a Dialogic extension. See <dialogstart>.
Section 9.6.2	<dialogend>	-	supported	
		attributes:		
		id	supported	
		mark	supported	
Section 9.6.3	<send>		supported	
		attributes:		
		event	supported	

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
		target	supported	If the target attribute is not specified or the target attribute value does not exist when using <send> to send an event, the event will be sent to "source" by default as documented in RFC 5707.
		namelist	supported	
Section 9.6.4	<exit>		supported	
		attributes:		
		namelist	supported	
Section 9.6.5	<disconnect>		supported	
		attributes:		
		namelist	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	

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
		barge	supported	No effect for video.
		cleardb	supported	
		offset	supported	No effect for video.
		skip	supported	
		xml:lang	supported	
		events:	-	
		pause	supported	
		resume	supported	
		forward	supported	
		backward	supported	
		restart	supported	
		toggle-state	supported	
		terminate	supported	
		shadow variables:	-	
		play.amt	supported	No support for video.
		play.end	supported	
Section 9.7.1.1	<audio>	-	supported	
		attributes:		
		uri	supported	Supports the following schemes: "file://", "http://".
		format	supported	Refer to Media File Formats.

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
		audiosamplerate	supported	Refer to Media File Formats.
		audiosamplesize	supported	Refer to Media File Formats.
		iterate	supported	
		tts	supported	
		xml:lang	supported	
Section 9.7.1.2	<video>	-	supported	
		attributes:		
		uri	supported	Supports the following schemes: "file://", "http://".
		format	supported	Refer to Media File Formats.
		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.

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
		framerate	supported	No values defined.
		iterate	supported	
		tts	supported	
Section 9.7.1.3	<media>	-	supported	
		attributes:		
		tts	supported	
Section 9.7.1.4	<var>	-	supported	
		attributes:		
		type	supported	
		subtype	supported	
		value	supported	
		tts	supported	
		xml:lang	supported	English (US) codes: "en-us", "eng-us", "eng". Chinese (PRC) codes: "zh-cn", "chi".
Section 9.7.1.5	<playexit>	-	supported	
Section 9.7.2	<dtmfgen>	-	supported	
		attributes:		
		id	supported	
		digits	supported	
		level	supported	
		dur	supported	
		interval	supported	

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
		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	

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
		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	
Section 9.7.4	<record>	-	supported	
		attributes:	-	
		id	supported	
		append	supported	Supports audio only.
		dest	supported	Uses dx_ device. Supports the following schemes: "file://", "http://".
		audiodest	supported	Uses mm_ device. Supports the following schemes: "file://", "http://".

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
		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	

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
		prespeech	supported	If no audio energy is detected for the amount of time specified by prespeech, the recording is terminated and automatically deleted.
		postspeech	supported	
		termkey	supported	No effect for video.
		events:	-	
		pause	not supported	
		resume	supported	
		toggle-state	not supported	
		terminate	supported	
		terminate.cancelled	supported	
		terminate.finalsilence	supported	
		nospeech	not supported	
		shadow variables:	-	
		record.len	supported	
		record.end	supported	
		record.recordid	not supported	
	<play> (child)	-	supported	See <play> definition.

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
	<tonegen> (child)	-	not supported	See <tonegen> definition.
Section 9.7.4.3	<recordexit>	-	supported	
Section 9.7.5	<dtmf>, <collect>	-	supported	
		attributes:	-	
		id	supported	
		cleardb	supported	Supports "true" only.
		fdt	supported	
		idt	supported	
		edt	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)	-	supported	See <play> definition.
Section 9.7.5.2	<pattern>	-	supported	

RFC 5707 Ref	Element Name	Attribute/Event/Shadow Variable Name	Level of Support	Comments
		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	
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

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	
Section 9.9.1.1	<voice>, <silence>, <tvoice>, <tsilence>	-	not supported	
		attributes:	-	
		len	not supported	

RFC 5707 Ref	Element Name	Attribute/Event Name	Level of Support	Comments
		sen	not supported	
Section 9.9.2	<gain>	-	supported	
		attributes:	-	
		id	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:	-	

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

Table 9. MSML Dialog Speech Package Support

RFC 5707 Ref	Element Name	Attribute/Event Name	Level of Support	Comments
Section 9.10.1	<speech>	-	supported	
		attributes:	-	

RFC 5707 Ref	Element Name	Attribute/Event Name	Level of Support	Comments
		id	supported	
		noint	supported	
		norect	supported	
		spcmplt	supported	
		spincmplt	supported	
		confidence	supported	
		sens	not supported	
		starttimer	supported	
		iterate	supported	
		events:	-	
		sens	not supported	
		starttimer	supported	
		terminate	supported	
		shadow variables:	-	
		speech.end	supported	
		speech.results	supported	
Section 9.10.1.1	<grammar>	-	supported	
		attributes:	-	
		uri	supported	
		iterate	supported	
Section 9.10.1.2	<match>	-	supported	
Section	<noinput>	-	supported	

RFC 5707 Ref	Element Name	Attribute/Event Name	Level of Support	Comments
9.10.1.3		**attributes:**	-	
		iterate	supported	
Section 9.10.1.4	<nomatch>	-	supported	
		attributes:	-	
		iterate	supported	
Section 9.10.1.5	<speechexit>	-	supported	
Section 9.10.2	<play>	-	supported	
Section 9.10.2.1	<tts>	-	supported	
		attributes:	-	
		uri	supported	
		iterate	supported	
		xml:lang	supported	

Table 10. MSML Dialog Fax Detection Package Support

RFC 5707 Ref	Element Name	Attribute/Event Name	Level of Support	Comments
Section 9.11.1	<faxdetect>	-	supported	
		attributes:	-	
		id	supported	
		events:	-	
		terminate	supported	
		shadow variables:	-	
		faxdetect.tone	supported	
		faxdetect.end	supported	
Section 9.11.2	<faxdetectexit>	-	supported	

Table 11. 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 12. 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 10 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	

RFC 5707 Ref	Element Name	State Parameters	Attribute Name	Level of Support	Comments
		dialog	-	supported	
		stream	-	supported	
		Child Elements			
Section 10.2.2	<auditresult>	-	-	supported	Based on Table 10 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	

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

Table 13. 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 10 Audit Core framework.
Section 10.3.1		sipdialog	-	supported	
		sipdialog.localseq	-	supported	
		sipdialog.remoteseq	-	supported	

RFC 5707 Ref	Element Name	State Parameters	Attribute Name	Level of Support	Comments
		sipdialog.localURI	-	supported	
		sipdialog.remoteURI	-	supported	
		sipdialog.remotetarget	-	supported	
		sipdialog.routeset	-	supported	
		localsdp	-	supported	
		remotesdp	-	supported	
		dialog	-	supported	
		stream	-	supported	
		Child Elements			
Section 10.3.2.1	<auditresult>	sipdialog	callid	supported	Based on Table 10 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	

RFC 5707 Ref	Element Name	State Parameters	Attribute Name	Level of Support	Comments
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 14. 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 10 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 10 Audit Core framework.

RFC 5707 Ref	Element Name	State Parameters	Attribute Name	Level of Support	Comments
			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 15. 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 10 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 10 Audit Core framework.
			media	supported	
			dir	supported	
			compressed	supported	
			display	not supported	

RFC 5707 Ref	Element Name	State Parameters	Attribute Name	Level of Support	Comments
			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	

MSML Support Details

The following sections provide details about MSML support.

MSML Core Package Support

For list of supported MSML Core Package elements, see MSML Core Package Support.

MSML Conference Core Package Support

For list of supported MSML Conference Core Package elements, see 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

<var>

specifies the generation of audio using prerecorded audio segments

<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.

Note: The change of going from active speakers to silence will not be reported.

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:

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

<audiomix>

Parent: <createconference>, modifyconference>, <destroyconference>

Child: <n-loudest>, <asn>

Note: The <audiomix> element cannot be destroyed using <destroyconference>. Instead the whole conference is destroyed.

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:

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

<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 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:

- Creating a basic audio conference with <asn> and <n-loudest>.
- Creating a four party layout video conference.
- Single party layout video conference using a <selector> element.

<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:

- Destroying a conference.

<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:

- Muting an audio stream flowing into a conference.
- Un-muting an audio stream flowing into a conference.

<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:

- Joining preferred party, full-duplex and listen-only parties to an audio conference.
- Call center coach-pupil conference.
- Un-joining streams using wildcards.
- Continuous digit collection on a conference participant.
- Creating a four party layout video conference.
- Expanding a four party layout to a six party layout video conference.
- Single party layout video conference using a <selector> element.
- Sequencing parties through regions in a video conference layout.

<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:

- Modifying a basic audio conference with <asn> and <n-loudest>.
- Expanding a four party layout to a six party layout video conference.
- Contracting a six party layout to a four party layout video conference.
- Layered regions in a video conference.

<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:

- Muting an audio stream flowing into a conference.
- Un-muting an audio stream flowing into a conference.
- Sequencing parties through regions in a video conference layout.

<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:

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

<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:

```
+----------+-----+
|          |  2  |
|        1 +-----+
|          |  3  |
+----+-----+-----+
| 6  | 5   |  4  |
+---+------+-----+

<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:

```
+---------- --+
| root        |
|background   |
|         +--- +--
|         |    |//
|         |foo |//
+--------+----+//
        |////////
```

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 RFC 5707, 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:

- Creating a four party layout video conference.
- Expanding a four party layout to a six party layout video conference.
- Contracting a six party layout to a four party layout video conference.
- Layered regions in a video conference.

<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, VGA, 720p, and 720p_4x3. The attribute is mandatory when creating the conference.

Events

None.

Shadow Variables

None.

Examples

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

- Creating a four party layout video conference.
- Expanding a four party layout to a six party layout video conference.
- Contracting a six party layout to a four party layout video conference.
- Layered regions in a video conference.
- Single party layout video conference using a <selector> element.

<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.

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:

- Single party layout video conference using a <selector> element.

<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 following attributes 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 when speech is present. 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". • A value of "true_enhanced" means that the stream MUST always be mixed independent of the speech detection algorithm. This setting is better suited for use cases such as providing music or a soundtrack as an input into the conference.
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".
dlgc:echo_cancel	Optional. This is a Dialogic extension. This attribute is used to enable or disable echo cancellation. • A value of "disable" may be specified. Default is "disable". • A value of "enable" may be specified.

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:

- Joining preferred party, full-duplex and listen-only parties to an audio conference.
- Call center coach-pupil conference.
- Setting <stream> attribute for echo cancellation.
- Muting an audio stream flowing into a conference.
- Un-muting an audio stream flowing into a conference.
- Un-joining streams using wildcards.
- Creating a four party layout video conference.
- Expanding a four party layout to a six party layout video conference.
- Single party layout video conference using a <selector> element.
- Sequencing parties through regions in a video conference layout.

<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:

- Un-joining streams using wildcards.
- Contracting a six party layout to a four party layout video conference.

<var>

Parent: <play>

Child Elements: None.

Description

The <var> element specifies the generation of audio using prerecorded audio segments that are selected and dynamically played in sequence based upon a specified variable. The variable may represent such items as date, time, and money.

Stringing together prerecorded selected audio segments allows an application better control of the "sound and feel" of the service provided to end users. It also provides very high audio quality and allows the variables to blend seamlessly with the surrounding audio segments.

Attributes

Attributes	Description
type	Mandatory. Specifies the type of variable. Values for type include: "date", "digits", "duration", "money", "month", "number", "silence", "time", and "weekday".
subtype	Mandatory. Specifies a clarification of type. Specific values of subtype supported depend upon the type attribute value specified.
value	Mandatory. Text that specifies what should be rendered for the type and subtype attributes.

Attributes	Description
xml:lang	Optional. This is the language tag that specifies the language to use when rendering and playing the variable. If not specified as a <var> element attribute, the xml:lang attribute specified for the <play> element will be used. If not specified for either the <var> element or the <play> element, the default language specified for the media server will be used. If a default language is not specified for the media server, the default will be English-US.

<var> Variable Types and Subtypes

Type	Description	
date	The value is spoken as a date in the form specified by the subtype.	
	value	The value is always specified as YYYYMMDD (per ISO 8601, International Date and Time Notation) YYYY: 1900-2999 MM: 01-12 DD: 01-31
	subtypes (mandatory)	
	mdy	Specifies month, day and year Example: 20021015 is spoken as "October Fifteenth Two Thousand Two"
	dmy	Specifies day, month and year Example: 20021015 is spoken as "Fifteen October Two Thousand Two"
	ymd	Specifies year, month and day Example: 20021015 is spoken as "Two Thousand Two October Fifteen"
digits	The value is spoken as a string of digits, one at a time, in the form specified by the subtype.	
	value	0-9
	subtypes (mandatory)	
	gen	Digits are spoken as generic digits, one at a time (one, five, zero) with no pause
	ndn	Digits are spoken with North American dialing phone number phrasing (NPA-NXX-XXXX), with appropriate pauses
duration	Duration is specified in seconds and is spoken in one or more units of time as specified by the subtype.	
	value	1 - 4,294,967,295 (>136 years)
	subtypes (mandatory)	
	yrs	The value is converted and spoken as years, days, hours, minutes, and seconds Example: 31626061 is spoken as "one year, one day, one hour, one minute, and one second"

Type	Description	
	hrs	The value is converted and spoken as hours, minutes, and seconds Example: 3661 is spoken as "one hour, one minute, and one second"
	mns	The value is converted and spoken as minutes and seconds Example: 3661 is spoken as "sixty one minutes, and one second"
money	Money is specified in the smallest unit of currency for the indicated subtype. The value is converted and spoken, per the subtype, as large units of currency followed by the remainder in smaller units of currency (for example, dollars and cents).	
	value	0 - 99999999999
	subtypes (mandatory)	
	usd	US dollar (cents) - (format: $$¢¢) Example: 1025 is spoken as "ten dollars and twenty five cents"
	cny	Chinese yuan (fen) - (format: $$¢¢) Example: 1255050 is spoken as "one wan, two thousand, five hundred, five shi, dollar, and fifty cents"
month	The value is spoken as a month and is specified in the MM format, with 01 denoting January, 02 denoting February, 10 denoting October, and so forth.	
	value	The value is always specified as MM: 01-12
	subtypes (optional) **Note:** If a subtype is included, the value must be "null".	
number	The value is a number in cardinal or ordinal form as specified by the subtype.	
	value	cardinal form: -999999999999999 to 999999999999999 ordinal form: 0 to 999999999999999
	subtypes (mandatory)	
	crd	cardinal 5111 is spoken as "five thousand, one hundred and eleven" 421 is spoken as "four hundred and twenty one"
	ord	ordinal 5111 is spoken as "five thousand, one hundred and eleventh" 421 is spoken as "four hundred and twenty-first"
silence	Plays a period of silence.	
	value	0 - 36000 (in 100 ms units up to 1 hour)
	subtypes (optional) **Note:** If a subtype is included, the value must be "null".	

Type	Description	
time	The value is spoken as a time of day in either twelve or twenty-four hour HHMM format according to ISO 8601, International Data and Time, as specified by the subtype.	
	value	The value is always specified as HHMM (per ISO 8601, International Date and Time Notation) HH: 00-24 refers to a zero-padded hour, 2400 (HHMM) denotes midnight at the end of the calendar day MM: 00-59 refers to a minute
	subtypes (mandatory)	
	t12	12 hour format Examples: 1730 is spoken as "five thirty p.m." 0530 is spoken as "five thirty a.m." 0030 is spoken as "twelve thirty a.m." 1230 is spoken as "twelve thirty p.m."
	t24	24 hour format 1700 is spoken as "seventeen hundred hours" Example: 2400 is spoken as "twenty four hundred hours"
weekday	The value is spoken as the day of the week. Days are specified as single digits, with 1 denoting Sunday, 2 denoting Monday, and so forth.	
	value	1 - 7 1 = Sunday 2 = Monday 3 = Tuesday 4 = Wednesday 5 = Thursday 6 = Friday 7 = Saturday)
	subtypes (optional) **Note:** If a subtype is included, the value must be "null".	
string	The value is a string of characters spoken as each individual character in the string. This type is a Dialogic extension.	
	value	a-Z, A-Z, 0-9, #, and * Example: "a34bc" is spoken as "A, three, four, B, C"
	subtypes (optional) Note: If a subtype is included, the value must be "null".	

Events

None.

Shadow Variables

None.

Examples

The following examples illustrate the usage of the <var> element:

- Playing the prompt for date.
- Playing the prompt for digits.
- Playing the prompt for duration.
- Playing the prompt for money.
- Playing the prompt for month.
- Playing the prompt for number.
- Playing the prompt for silence.
- Playing the prompt for time.
- Playing the prompt for weekday.
- Playing the prompt for string.

<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:

- Creating a four party layout video conference.
- Expanding a four party layout to a six party layout video conference.
- Contracting a six party layout to a four party layout video conference.
- Layered regions in a video conference.
- Single party layout video conference using a <selector> element.

MSML Dialog Core Package Support

For list of supported MSML Dialog Core Package elements, see MSML Dialog Core Package Support.

MSML Dialog Base Package Support

For list of supported MSML Dialog Base Package elements, see MSML Dialog Base Package Support.

MSML Dialog Group Package Support

For list of supported MSML Dialog Group Package elements, see MSML Dialog Group Package Support.

MSML Dialog Transform Package Support

For list of supported MSML Dialog Transform Package elements, see MSML Dialog Transform Package Support.

MSML Dialog Speech Package Support

For list of supported MSML Dialog Speech Package elements, see MSML Dialog Speech Package Support.

MSML Dialog Fax Detection Package Support

For list of supported MSML Dialog Fax Detection Package elements, see MSML Dialog Fax Detection Package Support.

MSML Audit Core Package Support

For list of supported MSML Audit Core Package elements, see MSML Audit Core Package Support.

MSML Audit Conference Package Support

For list of supported MSML Audit Conference Package elements, see MSML Audit Conference Package Support.

MSML Audit Connection Package Support

For list of supported MSML Audit Connection Package elements, see MSML Audit Connection Package Support.

MSML Audit Dialog Package Support

For list of supported MSML Audit Dialog Package elements, see MSML Audit Dialog Package Support.

MSML Audit Stream Package Support

For list of supported MSML Audit Stream Package elements, see MSML Audit Stream Package Support.

4. Deviations

Deviations from RFC 5707

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

The following is a list of deviations from RFC 5707:

- Nested groups are not supported.
- Only the "parallel" *topology* for <group> elements is supported.
- Wildcard ids for the <modifystream> element are not supported.
- Audio and video can play child elements but they can only be played sequentially. The <audio> and <video> elements must be child elements of the <media> element to play an audio-visual recording.
- The *offset* attribute of the <play> element has no effect when playing a video.
- 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.
- In the case of a <record> with a child <play>, the record does not start until the play is complete. Once the record begins, it can be terminated with the requested termkey.
- The *beep* attribute of the <record> element is supported and a proprietary implemenation.
- The <grammar> element fails if *xml:lang* attribute is not included. The <grammar> element is simply passed through to the speech server. It is the responsibility of the script author to ensure that the SRGS is valid and is processed correctly by the speech server.
- Media Server is ignoring the *audiosamplerate* and *audiosamplesize* attributes for a recording.
- When recording video using <record>, all video format parameters must be specified or none should be specified. If none specified, defaults from the stream being recorded will be used. A partial or incomplete format specification will not be accepted.

Differences between Native and Legacy MSML

PowerMedia XMS includes a Native distributed MSML service that leverages all of the technologies available in PowerMedia XMS.

The Native service is the default and recommended MSML service in PowerMedia XMS and replaces the previous legacy MSML service. The previous legacy MSML service is available for backward compatibility.

The following is a list of differences between Native and legacy MSML:

- Unlike Native MSML, the legacy MSML CPA (call progress analysis) primitive is not supported. Refer to Call Progress Analysis (CPA) Support for details on Native MSML CPA support.
- Conference Bridging is not supported.
- A connection (call) can only be joined to one other object at a time (call or conference).
- When playing media from a *http:// uri*, the web server must be configured to return the appropriate MIME type for the media.
- The codecs keyword of the *format* attribute for media play and record must be specified in the plural exactly as written in RFC 5707. The legacy MSML accepted codec as a substitute for codecs. This is no longer the case. The Native MSML strictly follows RFC 5707.
- The maxtime attribute of the <play> element refers to each media element and not the total time for the play.
- While the error codes from operation failures are the same, some of the descriptive text may have changed.
- DTMF pattern matching now works closer to RFC 5707 specification. Some script adjustments may be required.
- In Native mode, PowerMedia XMS has other active services besides MSML. In order to make sure that the incoming call is routed to the Native MSML service, the sip uri should begin with *sip:msml* or a route that matches the sip uri user part (which needs to be added to the call routing table in the Console). For more information, see the Routing section of *Dialogic® PowerMedia™ XMS Installation and Configuration Guide*.

- The MSML Media Server is engineered to take advantage of multi-core systems by processing concurrent calls simultaneously on different cores. When writing MSML scripts that include operations that affect more than one call id or affect a different call id than the id the script is executing on, the target call may be executing on a different processor core with its own independent timing characteristics.
  For example, when executing a script that joins two calls and a disconnect of one of the calls occurs simultaneously or immediately after the join script is sent to PowerMedia XMS, it is possible for the target call to complete the disconnect before the join can complete, thereby causing the join script to receive an error from PowerMedia XMS indicating that the target of the join does not exist.
- When an MSML <dialogEnd> is received by the media server: The Native MSML implementation will respond with an msml.dialog.exit event to the SIP call that created the MSML dialog; The legacy MSML implementation will respond with an msml.dialog.exit event to the SIP call that sent the MSML <dialogEnd>.

5. Feature Details

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:

- MSML Schema Validation
- Media Stream Direction Support
- HTTPS Play and Record
- Session File Transfer
- Pattern Matching with <dtmf>/<collect>
- Monitoring RTP Timeout Alarms
- DTMF Clamping for <record>
- Multiple URI for <audio> and <video>
- 3GP Multimedia Container
- Simultaneous Dual file (A+A/V) 3GP Record Mode
- Call Progress Analysis (CPA) Support

Note: A feature may not be supported on all platforms. For more information, see High-Level Feature Summary in Feature and Protocol Package Support.

MSML Schema Validation

MSML Schema Overview

The MSML specification consists of a set of XML schemas. The schemas may be used together, or any subset of schemas may be used for each MSML package. The schemas determine the level of support for MSML Packages and the elements and attributes defined within each MSML Package. The PowerMedia XMS includes an XML schema that has been modified by Dialogic. Because of the modified XML schema, elements and attributes have been added to the MSML Packages in order to provide additional levels of support for the PowerMedia XMS. An element or attribute that begins with "dlgc:" indicates that it is a Dialogic extension from the MSML specification. The following is an example of Dialogic extensions.

```
</xs:attribute>
  <xs:attribute ref="dlgc:conf_party_type"/>
  <xs:attribute ref="dlgc:echo_cancel"/>
  <!--xs:attribute name="dlgc:conf_party_type"type="xs:string"default="standard"use="optional">-->
<xs:attribute name="display"type="xs:string"/>
```

In addition, elements and attributes that are not supported by the modified XML schema have been restricted in the MSML Packages. The following is an example of a Dialogic restriction.

```
</xs:element>
<!--NOT SUPPORTED XS ELEMENT NAME="monitor">
  <xs:complexType>
  <xs:attribute name="id1" type="connID datatype" use="required"/>
  <xs:attribute name="id2" type="independentID datatype" use="required"/>
  <xs:attribute name="compressed" type="Boolean datatype" default="false"/>
  <xs:attribute ref="mark"/>
  </xs:complexType>
  </xs:element-->
```

```
</xs:element>
```

The Feature and Protocol Package Support section identifies the supported, extended, and unsupported MSML elements and attributes for the PowerMedia XMS. The schema for the PowerMedia XMS is located in the */etc/xms/msml* directory.

Enabling MSML Schema Validation

To enable MSML schema validation, navigate to the **MSML Configuration** page in the XMS web administration console and select the MSML Schema Validation parameter. The MSML Schema Validation parameter is disabled by default. Refer to the *Dialogic® PowerMedia™ XMS Installation and Configuration Guide* for details.

Note: Schema validation is inteded to be used during the script development process and disabled thereafter because it consumes additional CPU and memory resources, which can impact performance and system capacity.

Using Dialogic Elements and Attributes

When using an element or attribute from the Dialogic XML schema, the namespace tag "xmlns:dlgc=" must be included and followed by the pointer to DialogicTypes: "http://dialogic.com/DialogicTypes". Refer to the following example of the beginning of a MSML script that will use an element or attribute from the Dialogic XML schema.

```
<?xml version="1.0" encoding="UTF-8" ?>
<msml version="1.1" xmlns:dlgc="http://www.dialogic.com/DialogicTypes">
```

Note: If the namespace tag and the pointer to DialogicTypes are not included and the "dlgc" prefix is used, the MSML schema validation fails.

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.

INVITE without an SDP indicates to the MS to provide a full list of available audio and video codecs. This list would then be sent to the client which will decide which codecs to choose and notify back in answer SDP message. The MS would create a standard 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".

A control leg is created when an INVITE request is made from the application server (AS or MSML client) to the media server (MS) with no media connections and no media attributes within its SDP. This type of request is referred to as an INVITE with "NULL SDP." The following is an example of an INVITE request from an MSML client to the PowerMedia XMS with NULL SDP.

```
- Message Body
 - Session Description Protocol
    Session Description Protocol Version (v): 0
   + Owner/Creator, Session Id (0) 5646454564 645645564 IN IP4 0.0.0.0
Session Name (s): MSML Client
   + Connection Information (c): IN IP4 0.0.0.0
   + Time Description, active time (t): 0 0
```

A control leg consists of a SIP session/resource and an RTP session/resource, and it reserves a basic audio license resource. When establishing a control leg with the PowerMedia XMS, plan for these resources to be allocated and ensure that the PowerMedia XMS has sufficient licensed resources to fulfill this type of request.

For information on adding a license, refer to the *Dialogic® PowerMedia™ XMS Installation and Configuration Guide*.

HTTPS Play and Record

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

Feature Description

HTTPS 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 HTTPS server.

Requirements for HTTPS Support

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

Dialog Execution

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

The "https://" 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 HTTPS server. The HTTPS GET functionality is used to retrieve the information from the web server.

Audio and Video Playback

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

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 "https://" 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 HTTPS GET functionality will be used to retrieve the information from the web server. Currently, only the Dialogic proprietary video format is supported.

Audio and Video Recording

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

New audio files are stored in the location identified in the "https://" 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 "https://" scheme for the audiodest attribute, while the location of the new video file is stored in the location identified at "https://" scheme for the videodest attribute of the MSML <record> element. The HTTPS PUT functionality is used to send the information to the web server.

Session File Transfer

The <transfer> element is a proprietary Dialogic extension of RFC 5707 that supports transferring objects referenced by a source uri to a destination uri, including transferring file objects over HTTP and MSRP.

<transfer>

Parent: None.

Child Elements: <fileobj>, <transferstart>, <transferobjstart>, <transferobjdone>, <transferexit>

Description

The <transfer> element transfers objects defined by the child elements.

Transfer supports two states: transmit and suspend. Media transmission occurs in the transmit state and is suspended in the suspend state. The default initial state is transmit.

Attributes

Attributes	Description
id	An optional identifier that may be referenced elsewhere for sending events to the <transfer> element.

Attributes	Description
ftt	Defines the first transmit timer value. The first transmit timer is started when the transfer element is initially invoked or when the starttimer event is received. If the first transfer object completion has not been detected during this initial interval, the shadow variables are set and the dialog exits. Optional, default is 0s (wait forever for the first transfer object to complete).
itt	Defines the inter transmit timer. When specified, the timer is started when a transfer object completes. If a subsequent transfer object completion has not been detected when this timer expires, the shadow variables are set and the dialog exits. Optional, default is 0s (wait forever for the transfer object to complete).
mtt	Defines the maximum transmit timer. When specified, the timer is started when the transfer element processing begins. If the entire transfer operation completion has not been detected when this timer expires, the shadow variables are set and the dialog exits.
starttimer	Boolean value that defines whether the first transmit timer (ftt) is started initially. When set to false, the starttimer event must be received for it to start. Default is false.
retry	Specifies the number of times the transfer object elements execution may be retried upon failure, unless those elements specify differently. The value "forever" may be used to indicate that these may be retried any number of times. Default is 0.
iterate	Specifies the number of times the transfer object elements may be executed unless those elements specify differently. The value "forever" may be used to indicate that these may be executed any number of times. Default is 1.
initial	Defines the initial state for the transfer element. Default is "transmit".

Events

The following describes input events to the <transfer> element.

starttimer: starts the first transfer timer (ftt) if it has not already been started. Has no effect otherwise.

terminate: terminates the file transfer and assigns values to the shadow variables. When the destination is *file://* and the file transfer is interrupted, the local partial file will be automatically deleted.

resume: causes the transfer to enter transmit state.

Shadow Variables

transfer.duration: the cumulative duration of the individual transfer objects operation expressed as a duration in milliseconds.

transfer.end: contains the event that caused the transfer to stop. When the transfer stops because the objects have been completely transferred, end is assigned the value "transfer.complete". When the transfer stops due to timer expiration, *transfer.end* will be assigned one of the following: *transfer.failed.ftt*, *transfer.complete.itt*, or *transfer.complete.maxtime*.

Examples

The following examples illustrate the usage of the <transfer> element:

- PUT a file.
- Local file delete.
- Local file copy.
- Transfer file over MSRP and delete it (example in SIP INFO payload with content id).

<fileobj>

Parent: <transfer>

Child Elements: None.

Description

Defines one or more files to transfer from a source to a destination. The <fileobj> objects are processed sequentially. Subsequent <fileobj> may refer to products of previous <fileobj>. For example, the first <fileobj> appends file A to file B and the next <fileobj> transfers file B over MSRP to a file server.

Attributes

Attributes	Description
objid	An optional object id used in <transferobjdone> events if requested.
src	A whitespace separated list of uri to be transferred to the destination. Supported schemes include: *file://* and *xmsrp://*. Transferring multiple source uri to a single destination uri is context dependent. If the destination uri refers to a session based protocol such as MSRP, all specified files will be transferred within the same signaling session.
dest	The uri of the destination file. Supported schemes include: *file://* and *xmsrp://*.
contenttype	Mandatory. The MIME type of the content being transferred.
retry	The number of attempts to transfer the file before reporting an error.
iterate	The number of times to execute the file transfer.
delete	When set to true, will delete the file(s) referenced in the src after a successful transmission. The delete attribute is only supported for the *file://* uri scheme. Default: false. A simple file delete operation (without an actual transfer) can be accomplished by setting the delete attribute to true and omitting a destination uri.
append	When set to true, the source file(s) will be appended to the destination file if it exists. If the destination file does not exist, it will be created. The append operation is supported for audio media with *file://* scheme uri only. The source and destination files must have matching containers and media encoding. Default is false.
overwrite	When set to false, the *file://* destination will not be overwritten if it exists. The overwrite attribute is ignored if the destination scheme is not *file://*. Default is true.

Events

None.

Shadow Variables

None.

Examples

None.

<transferstart>

Parent: <transfer>

Child Elements: None.

Description

The <transferstart> child element requests that an event be sent when the transfer operation has begun. When triggered, the following will be executed:

```
<send target="source" event="transfer.start"/>
```

Attributes

None.

Events

None.

Shadow Variables

None.

Examples

None.

<transferobjstart>

Parent: <transfer>

Child Element: None.

Description

The <transferobjstart> child element requests that an event be sent when the transfer operation of an object has begun. When triggered, the following will be executed:

```
<send target="source" event="transfer.objstart" namelist="transfer.objid" />
```

Attributes

None.

Events

None.

Shadow Variables

None.

Examples

None.

<transferobjdone>

Parent: <transfer>

Child Elements: None.

Description

The <transferobjdone> child element is invoked when the transfer of each specified child object has been completed. The contents of this element may be used to send events.

Attributes

None.

Events

None.

Shadow Variables

None.

Examples

None.

<transferexit>

Parent: <transfer>

Child Elements: None.

Description

The <transferexit> child element must be invoked when the transfer of all specified objects has been completed. The contents of this element may be used to send events.

Attributes

None.

Events

None.

Shadow Variables

None.

Examples

None.

<fileop>

Parent: None.

Child Elements: None.

Description

The file operation to be executed on the specified file(s).

Note: If the **Allow Absolute Paths** parameter is set to "YES" on the **Media Configuration** page from the Console, file operation can be executed on any file on the operating system with root privileges (i.e., deleting an essential operating system file).

Attributes

Attributes	Description
id	An optional identifier for the operation.
operation	The operation to perform on the file(s). One of the following: copy, move, delete, append, or convert*.
src	The source uri for the file operation.
dest	The destination uri when the file operation is copy, move, or append.
overwrite	If set to false, the file operation will not overwrite the destination file if it exists. Default: true.
srcformat	Optional. The MIME type of the source content.
destformat	Optional. The MIME type of the destination content. **Note:** The destformat attribute can only be used for operations that have http:// scheme in the dest attribute.
*The convert operation supports conversion of PDF to/from TIFF file formats.	

Events

None.

Shadow Variables

None.

Examples

None.

MSRP File Transfer URI Scheme (xmsrp://)

The PowerMedia XMS specific *xmsrp://* uri scheme is used in conjunction with the <transfer> element to specify the MSRP specific parameters that are required to transfer an object using the MSRP protocol. The *xmsrp://* uri scheme has the following general format:

```
xmsrp://?offerer=sip_uri;answerer=sip_uri;file=filename;sigcontent=uri;sigheaders=uri
```

The *xmsrp://* uri parameters are defined as follows:

- offerer: The SIP uri of the offerer (local) endpoint.
- answerer: The SIP uri of the answerer (remote) endpoint.
- file: The path/filename of the file in the remote file system. If not supplied, or if specifying more than one file in the <fileobj> src, the filename from the source uri will be used.
- sigcontent: A cid: scheme uri (RFC 2392) that references content from the multipart/related body part (RFC 2387) that was received in the same INFO message as the MSML script itself. The referenced content will be attached as a body part to the SIP INVITE that will be used to set up the MSRP session for the transfer of the file uri(s) specified in the source attribute.
- sigheaders: A cid: scheme uri (RFC 2392) that references content from the multipart/related body part (RFC 2387) that was received in the same INFO message as the MSML script itself. The referenced content is expected to be a list of SIP headers and their values in the standard format:
    - Header-NameA: header value
    - Header-NameB: header value
- Alternatively, a relative *file://* uri can be specified that references a file on the PowerMedia XMS node. The file is relative to */etc/xms/msml*. For example, *file://headers.txt references /etc/xms/msml/headers.txt* on the PowerMedia XMS node.

If the *xmsrp://* uri is given as the source, the file is transferred from the answerer (remote) to the offerer (local) endpoint. If the *xmsrp://* uri is given as the destination, the file is transferred from the offerer (local) to the answerer (remote) endpoint.

Pattern Matching with <dtmf>/<collect>

DTMF input fulfills several roles within MSML dialogs. It is used to trigger events that will affect the media processing operation of other primitives. It is also used to collect DTMF digits from the media stream to be reported back to the source of the MSML dialog.

The following sections detail the supported pattern types and the general pattern matching rules:

Note: The reader should be familiar with the nominal reference specification RFC 5707 sections that describe <dtmf> and <pattern> elements.

Supported Patterns

The following pattern types are supported in PowerMedia XMS MSML:

- Exact: digits="123" The incoming digits must exactly match the pattern digits.
- Wildcard: digits="1x3" where "x" is the wildcard symbol representing any digit.
- Length: digits="length=3;cancel=*" exactly 3 digits must be entered or * cancels the <dtmf> and a dtmf.nomatch.cancel event is generated.
- MinMax: digits="min=1;max=3;rtk=#;cancel=*" between 1 and 3 digits must be entered followed by the rtk input termination digit #, * cancels the <dtmf> and a dtmf.nomatch.cancel event is generated. The default for min is 1 and for max is 50. At least one of min, max, rtk, or cancel must be specified.

Pattern Matching and Digit Buffer Rules

The following specifies the behavior for <dtmf>/<collect>/<pattern> elements:

- There is only one digit buffer per call shared by all primitives (<dtmf>/<collect>/<play>).
- The buffer is reset either during the call setup or through the cleardb attribute of <dtmf>/<collect>/<play>. The cleardb attribute defaults to false for <play> and true for <dtmf>.
- Each cleardb encountered resets the digit buffer; both in sequential and parallel (i.e., group topology) execution.
- A pattern can be in one of three states:
    - Non-matching: The initial state of a pattern.
    - Partially matching: At least one of the match conditions of the pattern is satisfied. For example, one or more (but not all) digits of a pattern match.
    - Matched: All match conditions of the pattern have been satisfied.
- The iterate attribute of the pattern, noinput, and nomatch elements specifies the number of times the element may be executed. An element is executed when its match conditions are satisfied and not necessarily by digit arrival. For example if using edt and rtk, all digits may match but the <pattern> is not executed until edt expires or the rtk digit is entered.
- Each pattern is independent and is comparing digits continuously as they arrive, staying in either the non-matching, partially matching, or matched state.
- If a pattern is in a partially matching state and a non-matching digit arrives, the pattern is reset to non-matching state and the non-matching digit is no longer considered for further matching of that pattern. After the pattern is reset, it resumes matching subsequent digits.
- The idt timer is stopped when a match or nomatch occurs. If iterate is greater than 1 and idt is being used, idt restarts when the next digit arrives.
- When using <pattern> with a digits attribute that contains an rtk key definition and contains MinMax definitions, the incoming digit is compared with the specified rtk digit before being compared with the rest of the pattern. When using a pattern that contains an rtk definition with patterns that do not contain MinMax definitions, the incoming digit is compared with the pattern digits before being compared with the defined rtk digit.
- If a MinMax pattern with rtk is being used with the edt timer and when the minimum digit condition is satisfied, the following is true:
    - The edt timer is started. Each successive digit will restart the edt timer until the maximum number of digits is collected.
    - If the rtk digit arrives before edt expiry, the pattern transitions immediately to the matched state and rtk digit is included in the dtmf.digits shadow variable.
    - When the maximum digit condition is satisfied, the arrival of a digit other than the rtk digit will cause the pattern to transition to the matched state. The digit buffer content (without the non-matching digit) is returned in the dtmf.digits shadow variable. The non-matching digit remains in the digit buffer for further matching.

- The expiry of the edt timer causes the pattern to transition to the matched state. The digit buffer content is returned in the dtmf.digits shadow variable.

- The terminating conditions of <dtmf>/<collect> are:
  - The corresponding iterate count of pattern, noinput or nomatch elements reaches zero.
  - The idt expires.
  - The edt timer expires. When using <dtmf>/<collect> with a defined edt timer, the edt timer starts once the pattern goes to a matched state. If no more digits arrive, the pattern process completes when the edt (or any shorter timer) expires.
  - The arrival of the cancel digit.

Monitoring RTP Timeout Alarms

RTP and RTCP timeout alarms can be monitored by enabling the parameters on the **MSML Advanced Configuration** page from the Console. For more information on how to configure, see the *Dialogic® PowerMedia™ XMS Installation and Configuration Guide*.

When the RTP and RTCP timeout alarms are enabled, PowerMedia XMS MSML will asynchronously send the following MSML event to the application server:

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
    <event name="alarm" id="conn:f11a5ac0-36f6a8c0-13c4-50022-64f-20cbb5db-64f">
        <name>alarm.rtp.timeout.state</name><value>on</value>
    </event>
</msml>
```

The supported alarm names are:

- alarm.rtp.timeout.state
- alarm.rtcp.timeout.state

The possible values are:

- on
- off

DTMF Clamping for <record>

The clamp attribute for <record> element can be used to enable DTMF clamping when recording.

The default behavior for <record> element is:

- If the termkey attribute is omitted from the <record> element, DTMF will be recorded to the file.
- If the termkey attribute is present, DTMF will be clamped from the recording.

The behavior with clamp attribute for <record> element is:

- The presence of this attribute overrides the default behavior.
- Valid values are "true" and "false".

  Note: If clamp="true" is used in conjunction with the prespeech or postspeech attributes, the DTMF will be clamped from the recording, but the DTMF signal will be treated as audio energy for processing the prespeech and postspeech attribute.

Multiple URI for <audio> and <video>

In order to minimize delays between multiple <audio> or <video> prompts, the uri attribute for <audio> and <video> elements can be used with multiple uri separated by space. The uri attribute identifies the location of the audio or video file.

The following example shows the format:

```
<audio uri="ur1 uri2 uri3">
```

The supplied file uri will be played sequentially with lower latency than if specified using multiple <audio> elements each with one uri.

3GP Multimedia Container

PowerMedia XMS MSML supports direct 3GPP file format (3GP) for both play and record operations. The 3GP container is specified for a subset of container characteristics, including restricted audio and video codecs defined for these network use cases. 3GP is an ISO-based multimedia file format and a subset of the MP4 file container.

PowerMedia XMS MSML supports 3GP record and playback through remote API interfaces that support video. Some of the highlighted functionality provided for 3GP container includes:

- Support for play and record directly to/from .3gp
- Support for audio only, video only, and multimedia (a/v) files
- Supported video codecs: H.264 (up to HD720p resolution at 2Mbps)

  Note: H.263 video codec is supported for play only; MPEG4 video codec is not supported in the initial release of this feature.
- Supported audio codecs: AMR-NB and AMR-WB

  Note: AAC codec is not supported in 3GP container.
- Supported DVR modes: skip forward, skip back, pause, resume (hint track required)
- Support for record with hint track to allow DVR modes on playback

For details on supported attributes, refer to the Media File Formats section.

Simultaneous Dual file (A+A/V) 3GP Record Mode

PowerMedia XMS MSML provides the capability to simultaneously record dual .3gp files with a single record operation. This special dual file record mode allows customers to record a separate audio only or video only .3gp file while simultaneously recording a multimedia .3gp file. This feature reduces the need for the application to do offline conversion into an audio only or video only format. A typical usage of this feature would be to record a simultaneous audio only version to send to a server to provide an audio transcript of a video message.

The following limitations apply to the dual file 3GP record:

- Both files must be .3gp format
- Audio only file must use same codec as multimedia file audio track
- Video only file must use same codec as multimedia file video track

Call Progress Analysis (CPA) Support

MSML supports call progress analysis (CPA) through the cpa=yes parameter in the SIP INVITE Request-URI or To header. Refer to the following example of a Request-URI or To header with the cpa=yes parameter:

```
msml@1.1.1.1;cpa=yes
```

The cpa=yes parameter triggers the following event internally, which enables CPA:

```
<send target="source" event="cpa" namelist="cpa.detect"/>
```

The CPA result is sent by XMS in a SIP INFO message:

```
</msml>
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
    <event name="cpa"
        id="conn:f3589b10-ddbaa8c0-13c4-65014-43e40-1f005b6d-43e40">
        <name>cpa.detect</name><value>voice</value>
</event>
</msml>
```

The cpa.detect values are as follows:

- "unknown"
- "answer-machine"
- "voice"
- "fax"
- "no-ringback"

6. Sample Use Case

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
- MSML Control Syntax in Use Case

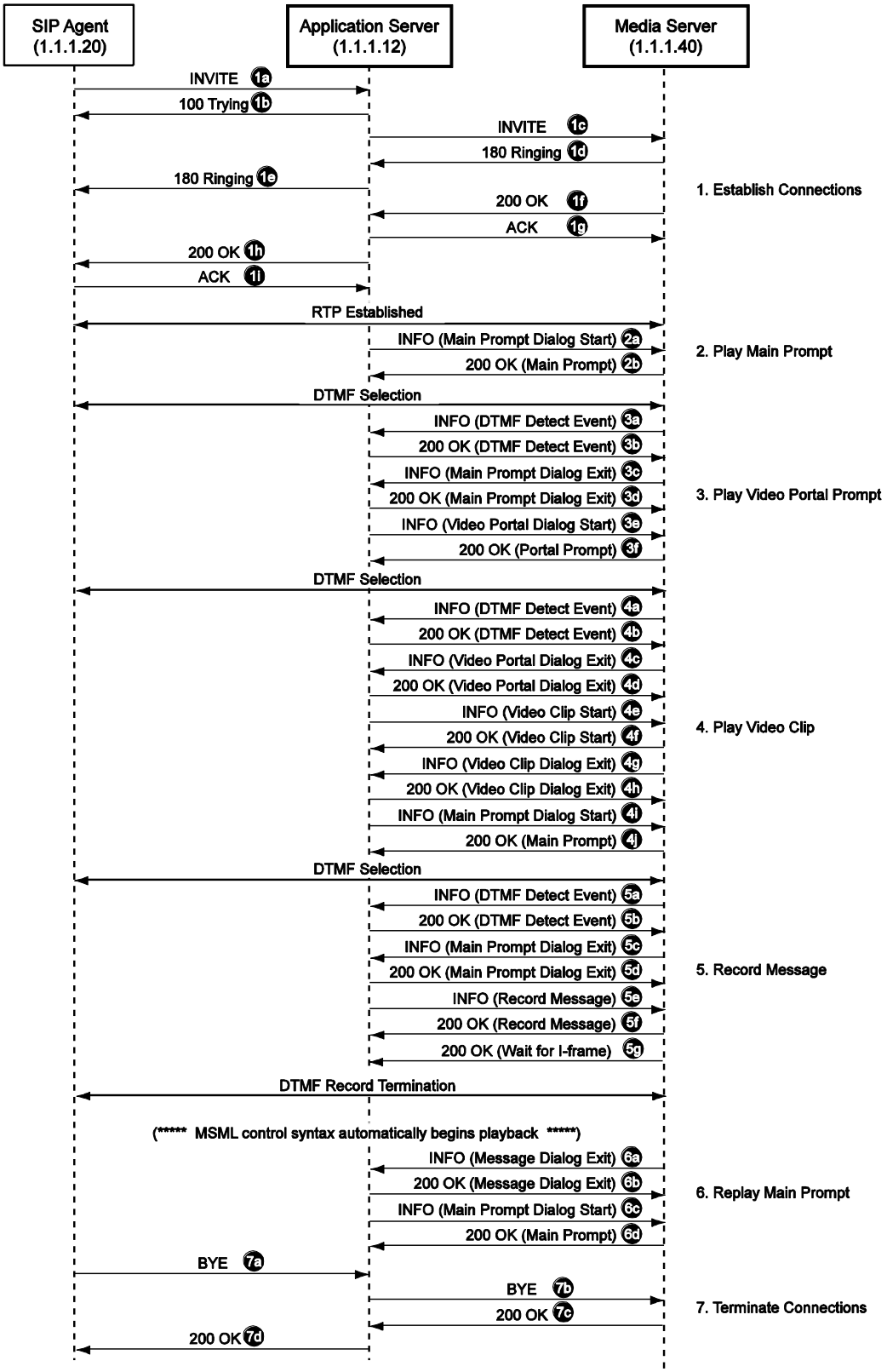
Use Case Description

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



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 Play Main Prompt) 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.

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. The following example shows the "To" header.

```
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 Play Main Prompt to Replay Main Prompt, the network connection identifier is shown in **bold** text.

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/moml+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>
```

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/moml+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>
```

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/moml+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/moml+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>
```

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/moml+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.

Note: The FF and RW will move to the nearest I-frame in the file. The effect of jumping to the nearest I-frame means that the FF and RW may not skip by the desired amount. For example, if you FF 3 seconds but the nearest I-frame is 20 seconds, it will FF 20 seconds instead.

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/moml+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>
```

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.

7. MSML Script Examples

This chapter provides sample MSML scripts. Topics include:

MSML Scripts for Audio Conferencing

The following sections provide examples of audio conferencing tasks.

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="1.1">
   <createconference name="1234" deletewhen="never">
       <audiomix id="mix1">
          <n-loudest n="3"/>
          <asn ri="10s"/>
       </audiomix>
   </createconference>
</msml>
```

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="1.1">
    <modifyconference id="conf:1234">
          <audiomix id="mix1">
             <n-loudest n="5"/>
             <asn ri="1s"/>
          </audiomix>
    </modifyconference>
</msml>
```

Joining preferred party, full-duplex and listen-only parties to an audio conference

The following example joins preferred party, full-duplex and listen-only parties to the audio conference created above in Creating a basic audio conference with <asn> and <n-loudest>. 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>
```

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" xmlns:dlgc="http://www.dialogic.com/DialogicTypes">
    <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>
```

Setting <stream> attribute for echo cancellation

The following examples show how to enable/disable echo cancellation for participants that are joined in a conference.

Setting <stream> attribute dlgc:echo_cancel = enable using <join>

```
<msml version="1.1" xmlns:dlgc="http://www.dialogic.com/DialogicTypes">
     <join id1="conn:jd87dfg4h" id2="conf:exampleConf ">
           <stream media="audio" dir="from-id1" dlgc:echo_cancel ="enable"/>
           <stream media="audio" dir="to-id1"/>
     </join>
</msml>
```

Setting <stream> attribute dlgc:echo_cancel = enable using <modifystream>

```
<msml version="1.1" xmlns:dlgc="http://www.dialogic.com/DialogicTypes">
     <modifystream id1="conn:1234" id2="conf:exampleConf">
           <stream media="audio" dir="from-id1" dlgc:echo_cancel="enable"/>
     </modifystream>
</msml>
```

Setting <stream> attribute dlgc:echo_cancel = disable using <modifystream>

```
<msml version="1.1" xmlns:dlgc="http://www.dialogic.com/DialogicTypes">
     <modifystream id1="conn:1234" id2="conf:exampleConf">
          <stream media="audio" dir="from-id1" dlgc:echo_cancel="disable"/>
     </modifystream>
</msml>
```

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>
```

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>
```

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>
```

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#_patterndetected" 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" interval="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

```
<?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>
```

Destroying a conference

This example destroys conference conf:1234.

```
<?xml version="1.0" encoding="UTF-8" ?>
<msml version="1.1">
     <destroyconference id="conf:1234"/>
</msml>
```

MSML Scripts for Video Conferencing

The following sections provide examples of video conferencing tasks.

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="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, and 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>
```

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 and 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>
```

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 and 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>
```

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>
```

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="1111" deletewhen="nomedia">
           <videolayout>
               <root size="CIF"/>
                   <selector id="switch1" method="vas">
                   <region id ="root"/>
                   </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>
```

Multiple party layout video conference using a <selector> element

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

Supporting <selector> in any region

```
<?xml version="1.0" encoding="UTF-8" ?>
<msml version="1.1">
     <createconference name="1111" deletewhen="nomedia">
           <videolayout type="text/msml-basic-layout">
               <root size="CIF"/>
                   <selector id="switch1" method="vas">
                   <region id="1" left="0" top="0" relativesize="2/3"/>
                   </selector>
                   <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>
      </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="1"/>
         <stream media="video" dir="to-id1"/>
     </join>
     <join id1="conn:0002" id2="conf:1111">
         <stream media="audio"/>
         <stream media="video" dir="from-id1" display="2"/>
         <stream media="video" dir="to-id1"/>
     </join>
     <join id1="conn:0003" id2="conf:1111">
         <stream media="audio"/>
         <stream media="video" dir="from-id1" display="3"/>
         <stream media="video" dir="to-id1"/>
     </join>
     <join id1="conn:0004" id2="conf:1111">
         <stream media="audio"/>
         <stream media="video" dir="from-id1" display="4"/>
         <stream media="video" dir="to-id1"/>
     </join>
</msml>
```

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.

Recording a segment of a video conference

```
<?xml version="1.0" encoding="UTF-8" ?>
<msml version="1.1">
    <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>
```

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>
```

Voice activated switching

The following examples show voice activated switching ("vas").

Creating a video conference

```
<?xml version="1.0" encoding="UTF-8" ?>
<msml version="1.1">
<createconference name="1234" 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>
```

Joining parties to the conference

```
<join id1="conn:1234" id2="conf:1234">
<stream media="audio"/>
<stream media="video" dir="from-id1" display="1"/>
<stream media="video" dir="to-id1"/>
</join>
</msml>
<?xml version="1.0" encoding="UTF-8" ?>
<msml version="1.1">
<join id1="conn:4321" id2="conf:1234">
<stream media="audio"/>
<stream media="video" dir="from-id1" display="2"/>
<stream media="video" dir="to-id1"/>
</join>
</msml>
```

Modifying the conference to six region layout with first region displaying active speaker

```
<?xml version="1.0" encoding="UTF-8" ?>
<msml version="1.1">
<modifyconference id="1234">
<videolayout>
<selector id="switch" method="vas">
<region id="1" left="0" top="0" relativesize="2/3"/>
</selector>
<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>
```

Restoring the conference back to static setting

```
</modifyconference>
</msml>
<?xml version="1.0" encoding="UTF-8" ?>
<msml version="1.1">
<modifyconference id="1234">
<videolayout>
<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>
</modifyconference>
</msml>
```

MSML Scripts Examples for <var> element

The following sections provide examples of using the <var> element.

Playing the prompt for date

Demonstrates playing the prompt for date with a subtype of "mdy".

```
<?xml version="1.0" encoding="UTF-8" ?>
      <msml version="1.1">
        <dialogstart target="conn:1234" type="application/moml+xml" name="sample">
           <play>
              <var type="date" subtype="mdy" value="20030601"/>
           </play>
           <send target="source"
                 event="done"
                 namelist="play.amt play.end"/>
         </dialogstart>
      </msml>
```

Playing the prompt for digits

Demonstrates playing the prompt for digits with a subtype of "gen".

```
<?xml version="1.0" encoding="UTF-8" ?>
      <msml version="1.1">
        <dialogstart target="conn:1234" type="application/moml+xml" name="sample">
           <play>
              <var type="digits" subtype="gen" value="9739676200" xml:lang="en-us"/>
           </play>
           <send target="source"
                 event="done"
                 namelist="play.amt play.end"/>
         </dialogstart>
      </msml>
```

Playing the prompt for duration

Demonstrates playing the prompt for duration with a subtype of "hrs".

```
<?xml version="1.0" encoding="UTF-8" ?>
      <msml version="1.1">
        <dialogstart target="conn:1234" type="application/moml+xml" name="sample">
           <play>
              <var type="duration" subtype="hrs" value="2563145"/>
           </play>
           <send target="source"
                 event="done"
                 namelist="play.amt play.end"/>
         </dialogstart>
      </msml>
```

Playing the prompt for money

Demonstrates playing the prompt for money with a subtype of "usd".

```
<?xml version="1.0" encoding="UTF-8" ?>
      <msml version="1.1">
        <dialogstart target="conn:1234" type="application/moml+xml" name="sample">
           <play>
              <var type="money" subtype="usd" value="0125"/>
           </play>
           <send target="source"
                 event="done"
                 namelist="play.amt play.end"/>
         </dialogstart>
      </msml>
```

Playing the prompt for month

```
<?xml version="1.0" encoding="UTF-8" ?>
      <msml version="1.1">
        <dialogstart target="conn:1234" type="application/moml+xml" name="sample">
           <play>
              <var type="month" subtype="" value="2"/>
           </play>
           <send target="source"
                  event="done"
                  namelist="play.amt play.end"/>
         </dialogstart>
      </msml>
```

Playing the prompt for number

Demonstrates playing the prompt for number with a subtype of "crd".

```
<?xml version="1.0" encoding="UTF-8" ?>
      <msml version="1.1">
        <dialogstart target="conn:1234" type="application/moml+xml" name="sample">
           <play>
              <var type="number" subtype="crd" value="511"/>
           </play>
           <send target="source"
                  event="done"
                  namelist="play.amt play.end"/>
         </dialogstart>
      </msml>
```

Playing the prompt for silence

```
<?xml version="1.0" encoding="UTF-8" ?>
      <msml version="1.1">
        <dialogstart target="conn:1234" type="application/moml+xml" name="sample">
            <play>
               <var type="silence" subtype="" value="5"/>
            </play>
            <send target="source"
                   event="done"
                   namelist="play.amt play.end"/>
          </dialogstart>
       </msml>
```

Playing the prompt for time

Demonstrates playing the prompt for time with a subtype of "t12".

```
<?xml version="1.0" encoding="UTF-8" ?>
      <msml version="1.1">
        <dialogstart target="conn:1234" type="application/moml+xml" name="sample">
            <play>
               <var type="time" subtype="t12" value="1750"/>
            </play>
            <send target="source"
                   event="done"
                   namelist="play.amt play.end"/>
          </dialogstart>
       </msml>
```

Playing the prompt for weekday

```
<?xml version="1.0" encoding="UTF-8" ?>
      <msml version="1.1">
        <dialogstart target="conn:1234" type="application/moml+xml" name="sample">
           <play>
              <var type="weekday" subtype="" value="2" xml:lang="eng"/>
           </play>
           <send target="source"
                  event="done"
                  namelist="play.amt play.end"/>
         </dialogstart>
      </msml>
```

Playing the prompt for string

```
<?xml version="1.0" encoding="UTF-8" ?>
      <msml version="1.1">
        <dialogstart target="conn:1234" type="application/moml+xml" name="sample">
           <play>
              <var type="string" subtype="" value="7adfHLL34kh" xml:lang="zh-cn"/>
           </play>
           <send target="source"
                  event="done"
                  namelist="play.amt play.end"/>
         </dialogstart>
      </msml>
```

MSML Scripts Examples for <transfer> element

The following sections provide examples of using the <transfer> element.

PUT a file

```
<msml version="1.1">
    <dialogstart target="conn:1234" type="application/moml+xml" name="12345">
        <transfer>
            <fileobj src="file://verification/verification_intro.wav"
            dest="http://server.example.com/media/verification_into.wav"/>
                <transferexit>
                    <send target="source" event="done"
                    namelist="transfer.duration transfer.end"/>
                </transferexit>
        </transfer>
    </dialogstart>
</msml>
```

Local file delete

```
<msml version="1.1">
    <dialogstart target="conn:1234" type="application/moml+xml" name="12345">
        <transfer>
            <fileobj delete="true" src="file://verification/verification_intro.wav"/>
        </transfer>
    </dialogstart>
</msml>
```

Local file copy

```
<msml version="1.1">
<dialogstart target="conn:1234" type="application/moml+xml" name="12345">
<transfer>
<fileobj src="file://verification/verification intro.wav"
dest="file://verification/copy_of_verification_intro.wav"/>
</transfer>
</dialogstart>
</msml>
```

Transfer file over MSRP and delete it (example in SIP INFO payload with content id)

```
Content-Type: Multipart/Related; boundary=example-1
         start="<950120.aaCC@XIson.com>";
         type="application/xml"

--example-1
Content-Type: application/xml
Content-ID: <950120.aaCC@XIson.com>

<msml version="1.1">
<dialogstart target="conn:1234" type="application/moml+xml" name="12345">
    <transfer>
        <fileobj objid="xyz" delete="true" src="file://verification/verification_intro.wav"
        dest="xmsrp://?offerer=userA@example.com;answerer=userB@example.com;
        file=verification intro.wav;sigcontent=cid:950120.aaCB@XIson.com"/>
            <transferobjdone/>
    </transfer>
</dialogstart>
</msml>

--example-1
Content-Type: application/MavVmExtensionData
Content-Description: Transparent data body
Content-Transfer-Encoding: base64
Content-ID: <950120.aaCB@XIson.com>

T2xkIE1hY0RvbmFsZCBoYWQgYSBmYXJtCkUgSS
BFIEkgTwpBbmQgb24gaGlzIGZhcm0gaGUgaGFk
IHNvbWUgZHVja3MKRSBJIEUgSSBPCldpdGggYS
BxdWFjayBxdWFjayBoZXJlLAphIHF1YWNrIHF1
YWNrIHRoZXJlLApldmVyeSB3aGVyZSBhIHF1YW
NrIHF1YWNrCkUgSSBFIEkgTwo=

--example-1--
```

8. Appendix A: Media Server Markup Language (MSML) Overview

This chapter provides a brief overview of the Media Server Markup Language (MSML). The descriptions are based on *Media Server Markup Language (MSML)*, RFC 5707 and are considered the nominal. Topics include:

Introduction

The Media Server Markup Language (MSML) is an Extensible Markup Language (XML) 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 RFC 5707.

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

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.

<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.

<send>

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

<result>

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

<event>

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

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.

<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.

<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.

<unjoin>

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

Conference Elements

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

<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.

<modifyconference>

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

<destroyconference>

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

Dialog Elements

Dialogs are used for interaction with a user.

<dialogstart>

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

The following attributes for <dialogstart> are Dialogic extensions of RFC 5707.

Attributes	Description
dlgc:target_display	Optional. This is a Dialogic extension. This is an attribute that allows a <dialogstart> element to be directed to a specific display region. This attribute is used only when the "target" attribute is set to be a conference connection (conf:xxxx). In this configuration, the attribute allows a video play media operation to be directed to a specific display region. If dlgc:target_display is not defined, the video will be played to a root of a conference.

<dialogend>

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

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.

Sending Events and Transaction Results to a Client

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.

Events

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

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)

Network Connections (conn)

Definition

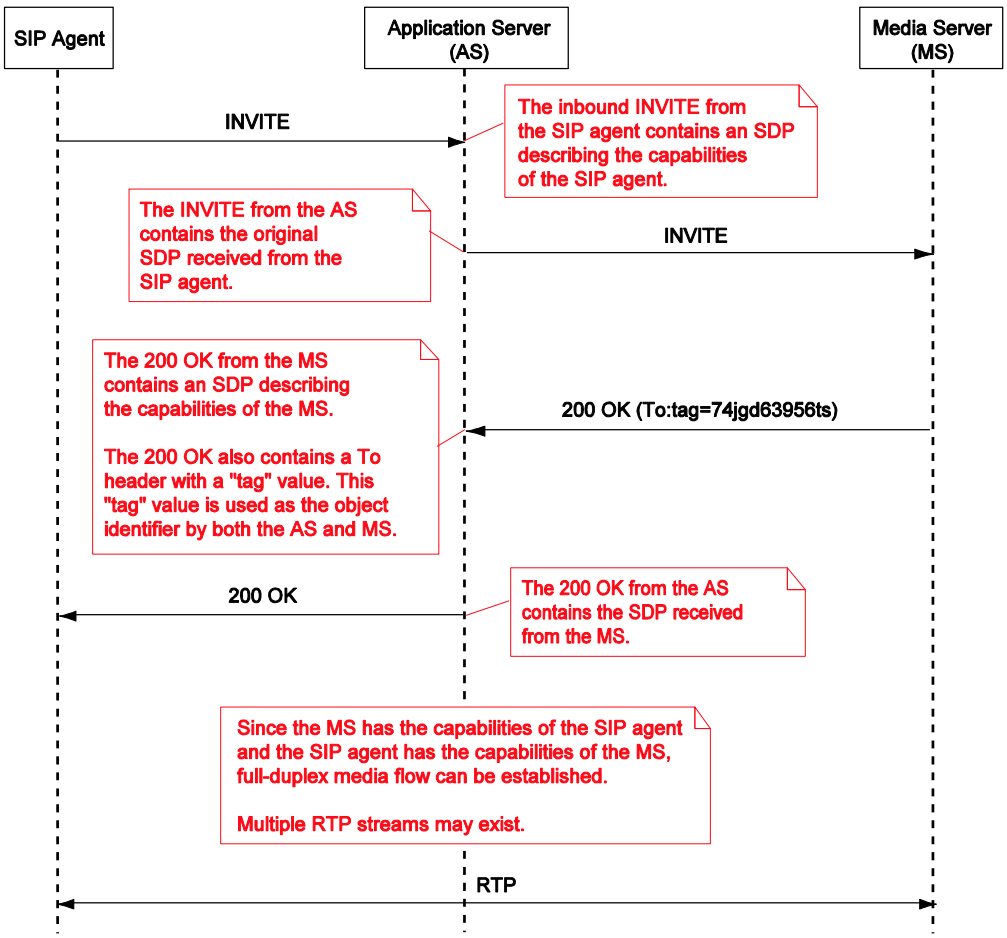
A Network Connection is an object or class defined in the MSML specification. Network Connection is an abstraction for the media processing resources involved in terminating the RTP session(s) of a call. For audio services, a connection instance presents a full-duplex audio stream interface within a 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".

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>
```

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, which 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>
```

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.

Media File Formats

The following section details the supported media containers and codecs. When selecting appropriate values for the <audio> or <video> element format attribute, refer to Media File Formats.

PowerMedia XMS has the ability to use the codec from the incoming stream(s) to record media into a container. From the MSML perspective, it is enabled by using codec=native for the containers that support codec=.

When playing media from a container that includes a header describing the media (AUD and WAV formats), the codecs=, audiosamplesize, and audiosamplerate attributes are not required and will be ignored if specified.

When playing media from a *http://* uri, the web server must be configured to supply the appropriate MIME type for the media in order for PowerMedia XMS to play the media. To select the appropriate MIME type for your media, refer to Media File Formats.

When recording audio and video, the filetype must be specified as video/proprietary. Since RFC 5707 only allows for one container specification, the audio container is inferred from the file extension. For the appropriate file extension specification for each supported audio container, refer to Media File Formats.

For example, the audio container (raw) is determined from the file extension of the audio destination:

```
<?xml version="1.0" encoding="UTF-8" ?>
<msml version="1.1">
   <dialogstart target="conf:exampleConf" type="application/moml+xml" name="DlgID">
      <record id="record1" maxtime="60s"
      audiodest=file:///root/mserver/record1.pcm videodest=file:///root/mserver/record1.vid
      format="video/proprietary;codecs=linear,native" audiosamplerate="8" audiosamplesize="16"/>
   </dialogstart>
</msml>
```

When recording video using <record>, all video format parameters must be specified or none should be specified. If none specified, defaults from the stream being recorded will be used. A partial or incomplete format specification will not be accepted.

Table 16. Media File Formats

	Play	Record	Default File Extension
Audio Container	WAV	WAV	*.wav
	AUD	AUD	*.aud
	Raw (headerless)	Raw (headerless)	*.vox *.pcm
	AMR	AMR	*.amr
	AMR-WB	AMR-WB	*.awb
	3GP	3GP	*.3gp
Video Container	VID	VID	*.vid
	3GP	3GP	*.3gp

Audio Play

Table 17. Audio Play - WAV

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
Linear	8 16	8000 11025 16000	audio/x-wav audio/wav audio/vnd.wave	n/a	n/a	n/a	audio/x-wav	n/a
Alaw	8	8000	audio/x-wav audio/wav audio/vnd.wave	n/a	n/a	n/a	audio/x-wav	n/a

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
Mulaw	8	8000	audio/x-wav audio/wav audio/vnd.wave	n/a	n/a	n/a	audio/x-wav	n/a

Table 18. Audio Play - Dialogic AUD (proprietary)

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
Linear	8 16	8000 11025 16000	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
Alaw	8	8000	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
Mulaw	8	8000	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-NB (4.75k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-NB (5.15k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-NB (5.90k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-NB (6.70k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-NB (7.40k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-NB (7.95k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-NB (10.20k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-NB (12.20k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-WB (6.6k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-WB (8.85k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-WB (12.65k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-WB (14.25k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-WB (15.85k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-WB (18.25k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
AMR-WB (19.85k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-WB (23.05k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a
AMR-WB (23.85k)	n/a	n/a	audio/proprietary audio/x-aud	n/a	n/a	n/a	audio/x-aud	n/a

Table 19. Audio Play - Raw (headerless)

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
Linear	8 16	8000 11025 16000	audio/vox	linear, pcm	8 16	8 11 16	audio/L8 audio/L16	rate=8000 rate=11025 rate=16000
Alaw	8	8000	audio/vox	alaw	8	8	audio/x-alaw-basic audio/PCMA	n/a
Mulaw	8	8000	audio/vox	mulaw	8	8	audio/basic audio/PCMU	n/a

Table 20. Audio Play - AMR

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
AMR	n/a	n/a	audio/amr	n/a	n/a	n/a	audio/AMR	n/a

Table 21. Audio Play - AMR-WB

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
AMR-WB	n/a	n/a	audio/amr-wb	n/a	n/a	n/a	audio/AMR-WB	n/a

Table 22. Audio Play - 3GP

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
AMR	n/a	n/a	audio/3gpp*	n/a	n/a	n/a	audio/3gpp	n/a
AMR-WB	n/a	n/a	audio/3gpp*	n/a	n/a	n/a	audio/3gpp	n/a
* To play a file with audio and video tracks, specify video/3gpp.								

Audio Record

Table 23. Audio Record - WAV

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
Linear	8 16	8000 11025 16000	audio/x-wav audio/wav audio/vnd.wave	linear, pcm	8 16	8 11 16	audio/x-wav	codec=L8 rate=8000 codec=L8 rate=11025 codec=L8 rate=16000 codec=L16 rate=8000 codec=L16 rate=11025 codec=L16 rate=16000
Alaw	8	8000	audio/x-wav audio/wav audio/vnd.wave	alaw	8	n/a	audio/x-wav	codec=alaw rate=8000
Mulaw	8	8000	audio/x-wav audio/wav audio/vnd.wave	mulaw	8	n/a	audio/x-wav	codec=mulaw rate=8000

Table 24. Audio Record - Dialogic AUD (proprietary)

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
Linear	8 16	8000 11025 16000	audio/proprietary audio/x-aud	linear, pcm	8 16	8 11 16	audio/x-aud	codec=L8 rate=8000 codec=L8 rate=11025 codec=L8 rate=16000 codec=L16 rate=8000 codec=L16 rate=11025 codec=L16 rate=16000
Alaw	8	8000	audio/proprietary audio/x-aud	alaw	8	8	audio/x-aud	codec=alaw rate=8000
Mulaw	8	8000	audio/proprietary audio/x-aud	mulaw	8	8	audio/x-aud	codec=mulaw rate=8000
AMR-NB (4.75k)	n/a	n/a	audio/proprietary audio/x-aud	amrnb_4_75k	n/a	n/a	audio/x-aud	codec=AMR mode=0

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
AMR-NB (5.15k)	n/a	n/a	audio/proprietary audio/x-aud	amrnb_5_15k	n/a	n/a	audio/x-aud	codec=AMR mode=1
AMR-NB (5.90k)	n/a	n/a	audio/proprietary audio/x-aud	amrnb_5_90k	n/a	n/a	audio/x-aud	codec=AMR mode=2
AMR-NB (6.70k)	n/a	n/a	audio/proprietary audio/x-aud	amrnb_6_70k	n/a	n/a	audio/x-aud	codec=AMR mode=3
AMR-NB (7.40k)	n/a	n/a	audio/proprietary audio/x-aud	amrnb_7_40k	n/a	n/a	audio/x-aud	codec=AMR mode=4
AMR-NB (7.95k)	n/a	n/a	audio/proprietary audio/x-aud	amrnb_7_95k	n/a	n/a	audio/x-aud	codec=AMR mode=5
AMR-NB (10.20k)	n/a	n/a	audio/proprietary audio/x-aud	amrnb_10_20k	n/a	n/a	audio/x-aud	codec=AMR mode=6
AMR-NB (12.20k)	n/a	n/a	audio/proprietary audio/x-aud	amrnb_12_20k	n/a	n/a	audio/x-aud	codec=AMR mode=7
AMR-WB (6.6k)	n/a	n/a	audio/proprietary audio/x-aud	amrwb_6_6k	n/a	n/a	audio/x-aud	codec=AMR-WB mode=0
AMR-WB (8.85k)	n/a	n/a	audio/proprietary audio/x-aud	amrwb_8_85k	n/a	n/a	audio/x-aud	codec=AMR-WB mode=1
AMR-WB (12.65k)	n/a	n/a	audio/proprietary audio/x-aud	amrwb_12_65k	n/a	n/a	audio/x-aud	codec=AMR-WB mode=2
AMR-WB (14.25k)	n/a	n/a	audio/proprietary audio/x-aud	amrwb_14_25k	n/a	n/a	audio/x-aud	codec=AMR-WB mode=3
AMR-WB (15.85k)	n/a	n/a	audio/proprietary audio/x-aud	amrwb_15_85k	n/a	n/a	audio/x-aud	codec=AMR-WB mode=4
AMR-WB (18.25k)	n/a	n/a	audio/proprietary audio/x-aud	amrwb_18_25k	n/a	n/a	audio/x-aud	codec=AMR-WB mode=5
AMR-WB (19.85k)	n/a	n/a	audio/proprietary audio/x-aud	amrwb_19_85k	n/a	n/a	audio/x-aud	codec=AMR-WB mode=6
AMR-WB (23.05k)	n/a	n/a	audio/proprietary audio/x-aud	amrwb_23_05k	n/a	n/a	audio/x-aud	codec=AMR-WB mode=7
AMR-WB (23.85k)	n/a	n/a	audio/proprietary audio/x-aud	amrwb_23_85k	n/a	n/a	audio/x-aud	codec=AMR-WB mode=8

Table 25. Audio Record - Raw (headerless)

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
Linear	8 16	8000 11025 16000	audio/vox	linear, pcm	8 16	8 11 16	audio/L8 audio/L16	rate=8000 rate=11025 rate=16000
Alaw	8	8000	audio/vox	alaw	8	8	audio/x-alaw-basic audio/PCMA	n/a

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
Mulaw	8	8000	audio/vox	mulaw	8	8	audio/basic audio/PCMU	n/a

Table 26. Audio Record - AMR

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
AMR	n/a	n/a	audio/amr	n/a	n/a	n/a	audio/AMR	mode=0-7 (default is 7)

Table 27. Audio Record - AMR-WB

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
AMR-WB	n/a	n/a	audio/amr-wb	n/a	n/a	n/a	audio/AMR-WB	mode=0-8 (default is 8)

Table 28. Audio Record - 3GP

Codec	Sample Size	Sample Rate	MSML Attribute format	MSML Attribute format codecs	MSML Attribute audiosamplesize	MSML Attribute audiosamplerate	XMS MIME Type	XMS MIME Parameters
AMR	n/a	n/a	audio/3gpp	n/a	n/a	n/a	audio/3gpp	codec=amr mode=0-7 (default is 7)
AMR-WB	n/a	n/a	audio/3gpp	n/a	n/a	n/a	audio/3gpp	codec=amr-wb mode=0-8 (default is 8)

Video Play

Table 29. Video Play - Dialogic VID (proprietary)

Codec	MSML Attribute format	MSML Attribute format codecs	XMS MIME Type	XMS MIME Parameters
h263	video/proprietary	n/a	video/x-vid	n/a
h263-1998	video/proprietary	n/a	video/x-vid	n/a

Codec	MSML Attribute format	MSML Attribute format codecs	XMS MIME Type	XMS MIME Parameters
h264	video/proprietary	n/a	video/x-vid	n/a
mp4v_es	video/proprietary	n/a	video/x-vid	n/a
jpeg	image/jpeg	jpeg	image/jpeg	n/a

Table 30. Video Play - 3GP

Codec	MSML Attribute format	MSML Attribute format codecs	XMS MIME Type	XMS MIME Parameters
h263	video/3gpp	n/a	video/3gpp	n/a
h263-1998	video/3gpp	n/a	video/3gpp	n/a
h264	video/3gpp	n/a	video/3gpp	n/a

Video Record

Table 31. Video Record - Dialogic VID (proprietary)

Video Codec	Resolution	Codecs	MSML Attribute profile	MSML Attribute level	MSML Attribute maxbitrate	MSML Attribute framerate	MSML Attribute imagewidth	MSML Attribute imageheight
H.263 / H.263-1998	CIF	h263	0	10	128000	10	352	288
	CIF		0	20	128000	15	352	288
	CIF		0	30	384000	30	352	288
	QCIF		0	10	128000	15	176	144
	QCIF		0	20	128000	30	176	144
	n/a	native	n/a	n/a	n/a	n/a	n/a	n/a
H.264	CIF	h264	0	1.2	384000	15	352	288
	CIF		0	1.3	384000	30	352	288
	CIF		0	1.3	768000	30	352	288
	CIF		0	2.0	384000	30	352	288
	CIF		0	2.1	384000	30	352	288
	CIF		0	2.2	768000	25	352	288
	HD720p		0	3.1	2000000	30	1280	720
	QCIF		0	1.0	40000	15	176	144
	QCIF		0	1.1	128000	15	176	144

Video Codec	Resolution	Codecs	MSML Attribute profile	MSML Attribute level	MSML Attribute maxbitrate	MSML Attribute framerate	MSML Attribute imagewidth	MSML Attribute imageheight
	QCIF		0	1.1	192000	30	176	144
	VGA		0	1.2	384000	15	640	480
	VGA		0	3.0	768000	25	640	480
	n/a	native	n/a	n/a	n/a	n/a	n/a	n/a
H.264 supports Baseline Profile (i.e., 66). This combined with level is used to set the profile_level_id. The level must be entered in "x.x" format (i.e., 10 and 1b are invalid).								
MPEG-4	CIF	mp4v-es	0	2	128000	15	352	288
	CIF		0	3	384000	15	352	288
	QCIF		0	0	44000	15	176	144
	QCIF		0	1	64000	15	176	144
	QVGA		0	3	384000	30	320	240
	VGA		0	4	800000	30	640	480
	n/a	native	n/a	n/a	n/a	n/a	n/a	n/a
MPEG-4 supports Simple Profile (i.e., SP3).								

Table 32. Video Record - 3GP

Video Codec	Resolution	Codecs	MSML Attribute profile	MSML Attribute level	MSML Attribute maxbitrate	MSML Attribute framerate	MSML Attribute imagewidth	MSML Attribute imageheight
H.263 / H.263-1998	CIF	h263	0	10	128000	10	352	288
	CIF		0	20	128000	15	352	288
	CIF		0	30	384000	30	352	288
	QCIF		0	10	128000	15	176	144
	QCIF		0	20	128000	30	176	144
	n/a	native	n/a	n/a	n/a	n/a	n/a	n/a
H.264	CIF	h264	0	1.2	384000	15	352	288
	CIF		0	1.3	384000	30	352	288
	CIF		0	1.3	768000	30	352	288
	CIF		0	2.0	384000	30	352	288
	CIF		0	2.1	384000	30	352	288
	CIF		0	2.2	768000	25	352	288
	HD720p		0	3.1	2000000	30	1280	720
	QCIF		0	1.0	40000	15	176	144

Video Codec	Resolution	Codecs	MSML Attribute profile	MSML Attribute level	MSML Attribute maxbitrate	MSML Attribute framerate	MSML Attribute imagewidth	MSML Attribute imageheight
	QCIF		0	1.1	128000	15	176	144
	QCIF		0	1.1	192000	30	176	144
	VGA		0	1.2	384000	15	640	480
	VGA		0	3.0	768000	25	640	480
	n/a	native	n/a	n/a	n/a	n/a	n/a	n/a
H.264 supports Baseline Profile (i.e., 66). This combined with level is used to set the profile_level_id. The level must be entered in "x.x" format (i.e., 10 and 1b are invalid).								