METREOS

IP Communications Applications for Business Advantage

# *Metreos MCE Framework API Reference Guide*

## *Metreos Communications Environment 2.1.4*

Information in this document is subject to change without notice.

# Licensing Terms

Use of this Software is subject to license restrictions. Carefully read this license agreement before using the software.

**This End User License Agreement (the "Agreement") is a legal agreement between you, either individually or as an authorized representative of the company or organization acquiring the license, and Metreos Corporation ("Metreos"). USE OF SOFTWARE INDICATES YOUR COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. If you do not agree to these terms and conditions, promptly return or, if electronically received, certify destruction of the Software and all accompanying items within five days after receipt of Software and you will receive a full refund of the applicable license fees paid.**

**1. License Grant**

a.The software programs you are installing, downloading, or have acquired with this Agreement, including any related equipment or hardware, documentation, updates, upgrades, modifications, revisions, copies and design data ("Software") are copyrighted, trade secret and confidential information of Metreos and its licensors who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Metreos grants to you, subject to payment of appropriate license fees, a non-exclusive, non-transferable, internal-use only, term license to use the Software owned or distributed by Metreos in machine readable, object-code form on the computer hardware or at the site(s) for which an applicable license fee has been paid, as authorized by Metreos.

b.To the extent that you design applications that operate on the Metreos Communications Environment (the "MCE"), Metreos grants you a non-exclusive, non-transferable, internal-use only, term license to use the Software, including the MCE, for the purposes of operating such programs, subject to payment of appropriate license fees.

c.The license granted herein is contingent on your (or your organization's) continued acceptance and subscription to the maintenance and support terms provided by Metreos. Termination by you or your organization of ongoing maintenance and support requirements terminates the license granted herein.

**2.Restrictions and Intellectual Property Ownership**

a.You may not (i) remove or modify any notice of Metreos' proprietary rights, (ii) re-license, rent, lease, timeshare, or act as a service bureau or provide subscription services for the Software, (iii) use the Software to provide third-party training, except for training agents and contractors authorized under this Agreement; (iv) assign this Agreement or give the Software or an interest in the Software to another individual or entity; (v) cause or permit reverse engineering or decompilation of the Software; (vi) disclose results of any Software benchwork tests without Metreos' prior written consent; or (vii) modify the Software or any portions thereof without Metreos' prior written consent.

b.The Software, which is copyrighted, and any modifications, upgrades, or updates thereto, is the sole and exclusive property of Metreos and is a valuable asset and trade secret of Metreos. Metreos retains all ownership and intellectual property rights to the Software and to any modifications, upgrades, or updates thereto. Except for the rights granted in herein above, you shall have no right, title, or interest of any kind in or to the Software.

c.Metreos may audit your use of the Software. If Metreos gives you or your organization reasonable advance written notice, you agree to cooperate with the audit, and to provide reasonable assistance and access to information. You agree to immediately remit any underpaid license and technical support fees determined as the result of such audit.

**3.Term and Termination.**

a. This Agreement remains effective until expiration or termination. This Agreement will immediately terminate upon notice if you exceed the scope of the license granted or otherwise fail to comply with the provisions in sections 1 and 2 above. For any other material breach of the Agreement, Metreos may terminate this Agreement if you are in breach and fail to cure the breach within thirty (30) days of written notification. If Software is provided for a limited term use, this Agreement will automatically expire at the end of the authorized term.

b.Upon termination of this Agreement for any reason, you shall within ten (10) business days return to Metreos all Software Additionally, you agree to delete from any permanent machine storage (i.e., hard disk) previously loaded copies of the Software in all forms. Upon request of Metreos, you shall certify in writing that all copies of the Software and associated documentation have been destroyed or returned to Metreos. The indemnity and limitation of liability obligations hereunder, as well as your obligations with respect to confidential treatment of the Software and Metreos' trade secrets, other intellectual property, and proprietary information, shall survive the termination of this Agreement.

**4. Limited Warranty.**

a.Metreos warrants that the Software will substantially operate as described in the applicable Software documentation for ninety (90) days after Metreos delivers it to you. **THE WARRANTY HEREIN IS EXCLUSIVE AND IN LIEU OF ALL OTHER WARRANTIES, WHETHER EXPRESS OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. METREOS MAKES NO WARRANTY THAT ANY SOFTWARE WILL PERFORM ERROR-FREE OR UNINTERRUPTED, OR THAT ALL ERRORS THEREIN CAN OR WILL BE CORRECTED. METREOS FURTHER DISCLAIMS ANY IMPLIED WARRANTIES ARISING FROM COURSE OF PERFORMANCE, COURSE OF DEALING OR USAGE OF TRADE.**

b.For any breach of the above warranty, Metreos' entire liability and your exclusive remedy shall be, at Metreos' option, either (1) refund of the fees paid upon return of the Software to Metreos, or (2) correction or replacement of the Software that does not meet this limited warranty, provided you have complied with the terms of this Agreement.

**5.Indemnity**

a.Metreos will defend and indemnify you against a claim that any Software, infringes a patent or copyright, provided that: (i) you notify Metreos in writing within thirty (30) days of the claim; (ii) Metreos has sole control of the defense and all related settlement negotiations; and (iii) you provide Metreos with the assistance, information, and authority reasonably necessary to perform the above; reasonable out-of-pocket expenses incurred by you in providing such assistance will be reimbursed by Metreos.

b.Metreos shall have no liability for any claim of infringement resulting from: (i) your use of a superseded or altered release of the Software if infringement would have been avoided by the use of a subsequent unaltered release of the Software which Metreos provides to you; or (ii) any information, design, specification, instruction, software, data, or material not furnished by Metreos or (iii) any combination of the Software with other hardware, software or processes that, but for the combination, the Software would not be infringing.

c.In the event that some or all of the Software is held or is believed by Metreos to infringe, Metreos shall have the option, at its expense: (i) to modify the Software to be non-infringing; or (ii) to obtain for you a license to continue using the Software. If it is not commercially feasible to perform either of the above options, then Metreos may require from you return of the infringing Software and all rights thereto. Upon return of the infringing Software to Metreos, you may terminate the Agreement with ten (10) days' written notice and you shall be entitled to a pro-rata refund of the fees paid for the infringing Software. This subsection sets forth Metreos' entire liability and exclusive remedy for infringement.

d.You will defend and indemnify Metreos and its licensors against any claim incurred by, borne by or asserted against Metreos or its licensors that relates to or results from (i) your use of the Software, (ii) any intentional or willful conduct or negligence by you or (iii) any breach of an applicable representation, covenant or warranty contained herein.

e.Should the party seeking indemnification ("Indemnitee") reasonably determine that the party from whom indemnity is sought ("Indemnitor") has failed to assume the defense of any claim referenced herein, Indemnitee shall have the right to assume such defense and have all expense and cost of the defense reimbursed by Indemnitor, including reasonable attorneys fees.

**6.Confidentiality**

The Software contains proprietary and confidential information of Metreos as well as trade secrets owned by Metreos. You agree to hold the Software in strict confidence and not to disclose the Software in any way except as expressly permitted hereunder. You agree to protect the Software at least to the same extent that you protect your similar confidential information, but in no event less than reasonable care. You further agree that you will not, directly or indirectly, copy the structure, sequence, or organization of the Software, nor will you copy any portion of the Software or related documentation to produce software programs that are substantially similar to the Software.

**7. LIMITATION OF LIABILITY**

**EXCEPT WHERE THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, IN NO EVENT WILL METREOS BE LIABLE FOR ANY INDIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES, OR DAMAGES FOR LOSS OF PROFITS, REVENUE, DATA, OR USE, INCURRED BY YOU OR ANY THIRD PARTY, WHETHER IN AN ACTION IN CONTRACT OR TORT, EVEN IF METREOS OR ANY OTHER PERSON HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGESMETREOS' LIABILITY FOR DAMAGES HEREUNDER SHALL IN NO EVENT EXCEED THE AMOUNT OF FEES PAID BY YOU FOR THE SOFTWARE OR SERVICE GIVING RISE TO THE CLAIM. IN THE CASE WHERE NO AMOUNT WAS PAID, METREOS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER.**

**8.Assignment; Jurisdiction.**

This Agreement will be binding upon, and will inure to the benefit of, the permitted successors and assigns of each party hereto. You may not assign, delegate, transfer, or otherwise convey this Agreement, or any of its rights hereunder, to any entity without the prior written consent of Metreos, and any attempted assignment or delegation without such consent shall be void. This Agreement, and all matters arising out of or relating to this Agreement, shall be governed by the laws of the State of Texas, United States of America. Any legal action or proceeding relating to this Agreement shall be instituted in any state or federal court in Travis or Dallas County, Texas, United States of America. Metreos and you agree to submit to the jurisdiction of, and agree that venue is proper in, the aforesaid courts in any such legal action or proceeding.

**9.Severability; Waiver**

In the event any provision of this Agreement is held to be invalid or unenforceable, the remaining provisions of this Agreement will remain in full force. The waiver by either party of any default or breach of this Agreement shall not constitute a waiver of any other or subsequent default or breach. Except for actions for nonpayment or breach of either party's intellectual property rights, no action, regardless of form, arising out of this Agreement may be brought by either party more than two years after the cause of action has accrued. The headings appearing in this Agreement are inserted for convenience only, and will not be used to define, limit or enlarge the scope of this Agreement or any of the obligations herein.

**10.Restricted Rights Notice.**

The Software is commercial in nature and developed solely at private expense. The Software is delivered as "Commercial Computer Software" as defined in DFARS 252.227-7014 (June 1995) or as a commercial item as defined in FAR 2.101(a) and as provided with only such rights as are provided in this Agreement, which is Metreos' standard commercial license for the Software. Technical data is provided with limited rights only as provided in DFAR 252.227-7015 (Nov. 1995) or FAR 52.227-14 (June 1987), whichever is applicable.

# 1. About This Guide

This section identifies the intended audience for this guide, explains how the guide is organized and lists the typographical conventions in effect.

## 1.1. Intended Audience

This guide is intended for use by programmers wishing to learn how to develop telephony applications for the Metreos Communications Environment. You should be familiar with the basics of the Metreos Visual Designer as a prerequisite to this document.

## 1.2. Organization of this Guide

This guide is contains the definition of every action and event in the Metreos Application Framework.

## 1.3. Typographical Conventions

The following typographical components are used for defining special terms and command syntax:

| Convention | Description |
|---|---|
| **Bold** typeface | Represents literal information such as<br>• Information and controls displayed on screen, including menu options, windows dialogs and field names<br>• Commands, file names, and directories<br>• In-line programming elements, such as class names and XML elements when referenced in the main text |
| *Italic* typeface | Italics typeface is used to denote<br>• A new concept<br>• A variable element such as *filename***.mca**. In this example, *filename* represents the filename and **.mca** is the extension.<br>• A reference to a chapter or section heading |
| `Courier` typeface | Denotes code or code fragments |
| UPPERCASE | Denotes keys and keystroke combinations such as CTRL+ALT+DEL. |

# TABLE OF CONTENTS

# 2. API REFERENCE

## 2.1. Call Control

### 2.1.1. Actions

#### Metreos.CallControl.AcceptCall Action

Accepts a phone call.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Return Value**

Returns `Success` if the call could be accepted; otherwise, `Failure` is returned.

**Remarks**

Acts as a provisional acceptance of an incoming call. To accept a call means to allow the call to be given the chance to be answered (or not).

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

# Metreos.CallControl.RejectCall Action

Rejects a phone call.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Return Value**

Returns `Success` if the call could be rejected; otherwise, `Failure` is returned.

**Remarks**

Rejects a call before it has a chance to be answered.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

# Metreos.CallControl.AnswerCall Action

Answers a phone call.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| DisplayName | System.String | Free string used to describe the recipient of the call |
| MmsId | System.String | ID of the specific media server to use. Use of this field will disable media server load-balancing features for this call. If this specified call ID is not valid, the AnswerCall will fail. |
| WaitForMedia | System.String | Indicates that the asynchronous response should not be sent until media has been established. The value of this field will affect the media values returned in the response. Valid values are TxRx (default), Tx, Rx, or None. |
| Conference | System.Boolean | Indicates that this call is to be placed in a conference. |
| ConferenceId | System.String | The ID of the conference to which the call should be added. Specify 0 if this is the first party in the conference. This field is ignored if Conference is false. |
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail. |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| MmsId | System.String | ID of the media server handling media for this call |
| ConnectionId | System.String | ID of the connection to the media server handling this media for this call |
| ConferenceId | System.String | ID of the conference on the media server that this connection was placed in. Valid only if Conference was specified as true. |

| | | |
|---|---|---|
| MediaTxIp | System.String | The IP address to which media is being sent to |
| MediaTxPort | System.Uint32 | The port that media is being sent to |
| MediaTxCodec | System.String | The codec used on the send channel |
| MediaTxFramesize | System.Uint32 | The framesize used on the send channel |
| MediaRxIP | System.String | The local IP address that media is being received on |
| MediaRxPort | System.Uint32 | The local port that media is being received on |
| MediaRxCodec | System.String | The codec used on the receive channel |
| MediaRxFramesize | System.Uint32 | The framesize used on the receive channel |

**Return Value**

Returns `Success` if the call could be answered; otherwise, `Failure` is returned.

**Remarks**

Attempts to answer a call, while provisioning a connection on a media server for this call. The AnswerCall action can use a new or existing conference in which to place the media server connection.

For two-way audio, the `WaitForMedia` field can either be left blank or set explicitly to `TxRx`. If, however, you need to play media only to a given connection, it is valid to specify `Rx`.

`MmsId` specifies which media server was used to create the connection for this call. In an environment containing multiple media servers clustered to one Application Runtime Environment, `MmsID` can be important for some applications. Creating a conference, for example, requires that all connections reside on the same media server. Using the `MmsId` parameter from the first connection placed into conference, you could then specify the same media server ID for the rest of the connections you want to conference. Doing so in this example is necessary because a connection is not guaranteed to be on the same media server as a different connection—even in the same script—unless the `MmsId` action parameter is specified.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

## Metreos.CallControl.MakeCall Action

Makes an outbound call from the Application Runtime Environment to a specified destination

**Operation**

Asynchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| To | System.String | The destination number for this call |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| From | System.String | The calling number, or caller ID, for this call |
| DisplayName | System.String | Free string describing the initiator of the call |
| MmsId | System.String | ID of the media server handling media for this call |
| PeerCallId | System.String | The call ID of the outstanding incoming call to be linked to this one in a peer-to-peer fashion |
| WaitForMedia | System.String | Indicates the asynchronous response should not be sent until media has been established. The value of this field will affect the media values returned in the response. Valid values are TxRx (default), Tx, Rx, or None. |
| ConnectionId | System.String | ID of the connection to the media server handling this media for this call |
| ConferenceId | System.String | ID of the conference on the media server in which this connection was placed. Valid only if Conference was specified as true. |
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Return Value**

Returns `Success` if the call could be made; otherwise, `Failure` is returned.

**Remarks**

Attempts to make a call, while provisioning a connection on the media server for this call. MakeCall can use a new or existing conference in which to place the connection to a media server.

For two-way audio, the `WaitForMedia` field can either be left blank or set explicitly to `TxRx`. If, however, you need only to play media to a specific connection, it would be valid to specify `Rx`.

`MmsId` specifies which media server was used to create the connection for this call. In an environment containing multiple media servers clustered to one Application Runtime Environment, `MmsID` can be important for some applications. Creating a conference, for example, requires that all connections reside on the same media server. Using the `MmsId` parameter from the first connection placed into conference, you could then specify the same media server ID for the rest of the connections you want to conference. Doing so in this example is necessary because a connection is not guaranteed to be on the same media server as a different connection—even in the same script—unless the `MmsId` action parameter is specified.

Other than the `CallId`, all pertinent information returned by this action is sent through the asynchronous success or failure event that occurs once the call has been made or failed.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

## Metreos.CallControl.Hangup Action

Terminates the established call with the specified ID. If two calls were established as peers and they have not since been bridged through a media server, both calls will be terminated when a Hangup is issued with either of their call IDs

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

**Return Value**

Returns `Success` if the system was able to hang up the call; if not, `Failure` is returned.

**Remarks**

Attempts to hang up a call. When the phone is hung up, the media server connection created for this call will be destroyed.

If, however, a conference was made automatically by MakeCall or AnswerCall, it will not delete the conference under any condition. This requirement is by design; the media server will automatically destroy any conference from which all participants leave.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

## Metreos.CallControl.BridgeCalls Action

Renegotiates media for two calls with peer-to-peer media streams so that the media streams are hair-pinned through the media server.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| MmsId | System.String | ID of the media server handling media for this call |
| ConnectionId | System.String | ID of the connection to the media server handling this media for this call |
| MediaTxIP1 | System.String | The IP address that media for connection 1 is being sent to |
| MediaTxPort1 | System.UInt32 | The port that media for connection 1 is being sent to |
| MediaTxCodec1 | System.String | The codec used for connection 1 on the send channel |
| MediaTxFramesize1 | System.UInt32 | The framesize used for connection 1 on the send channel |
| MediaRxIP1 | System.String | The IP address that media for connection 1 is being received from |
| MediaRxPort1 | System.UInt32 | The port that media for connection 1 is being received from |
| MediaRxCodec1 | System.String | The codec used for connection 1 on the receive channel |
| MediaRxFramesize1 | System.UInt32 | The framesize used for connection 1 on the receive channel |
| ConnnectionId2 | System.String | The new connection ID created for the peer of the call specified |
| MediaTxIP2 | System.String | The IP address that media for connection 2 is being sent to |

| | | |
|---|---|---|
| MediaTxPort2 | System.UInt32 | The port that media for connection 2 is being sent to |
| MediaTxCodec2 | System.String | The codec used for connection 2 on the send channel |
| MediaTxFramesize2 | System.UInt32 | The framesize used for connection 2 on the send channel |
| MediaRxIP2 | System.String | The IP address that media for connection 2 is being received from |
| MediaRxPort2 | System.UInt32 | The port that media for connection 2 is being received from |
| MediaRxCodec2 | System.String | The codec used for connection 2 on the receive channel |
| MediaRxFramesize2 | System.UInt32 | The framesize used for connection 2 on the receive channel |

**Return Value**

Returns `Success` if the call could be made; otherwise, `Failure` is returned.

**Remarks**

Use the port returned to identify whether a particular channel has been established. An unestablished channel will have a port value of 0.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

## Metreos.CallControl.UnbridgeCalls Action

Renegotiates media for two calls which have been established with the media server so that the media path is peer-to-peer.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | A unique token used to identify this particular call leg in actions and events |
| CallId2 | System.String | A unique token used to identify this particular call leg in actions and events |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

**Return Value**

Returns `Success` if the calls could be unbridged; otherwise, `Failure` is returned.

**Remarks**

The connection IDs associated with these calls will be defunct when this operation completes successfully.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

## Metreos.CallControl.BlindTransfer Action

Transfers an established call to the specified party.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | A unique token used to identify this particular call leg in actions and events |
| To | System.String | The dialed number to redirect this call to |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

**Return Value**

Returns `Success` if the call could be transferred; otherwise, `Failure` is returned.

**Remarks**

The specified call ID and its corresponding connection ID are defunct after this operation has completed successfully. *This feature is currently not supported for peer-to-peer calls.*

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

## Metreos.CallControl.BeginConsultationTransfer Action

Places the specified call on hold and makes a call to a third party.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | A unique token used to identify this particular call leg in actions and events |
| To | System.String | The dialed number of the person to contact regarding the transfer |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| TransferCallId | System.String | The call ID of the party being consulted |
| MmsId | System.String | ID of the media server handling media for this call |
| ConnectionId | System.String | ID of the connection to the media server handling this media for this call |

**Return Value**

Returns `Success` if the call could be made; otherwise, `Failure` is returned.

**Remarks**

*This feature is currently not supported for peer-to-peer calls*.

**Requirements**

Metreos Framework version 2.0

Metreos JTAPI Provider version 2.0

## Metreos.CallControl.EndConsultationTransfer Action

Completes the consultation transfer.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | A unique token used to identify this particular call leg in actions and events |
| TransferCallId | System.String | The call ID of the party being transferred |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

**Return Value**

Returns `Success` if the call could be made; otherwise, `Failure` is returned.

**Remarks**

All call IDs involved in this action and their corresponding connection IDs are invalid after this operation has completed successfully. *This feature is currently not supported for peer-to-peer calls*.

**Requirements**

Metreos Framework version 2.0

Metreos JTAPI Provider version 2.0

## Metreos.CallControl.Redirect Action

Re-routes a pending incoming call which has not yet been connected.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | A unique token used to identify this particular call leg in actions and events |
| To | System.String | The dialed number of the person to contact regarding the transfer |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

**Return Value**

Returns `Success` if the call could be made; otherwise, `Failure` is returned.

**Remarks**

The specified call ID is defunct after this operation has completed successfully.

**Requirements**

Metreos Framework version 2.0

Metreos JTAPI Provider version 2.0

### 2.1.2. Events

## Metreos.CallControl.IncomingCall Event

Indicates that a call has been received by the Application Runtime Environment.

**Type**

Triggering

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | Token used to identify this call |
| From | System.String | The calling number, or caller ID, for this call |
| To | System.String | The dialed number for this call |
| OriginalTo | System.String | The original number which was dialed. This will be the same as the To field unless the call has been redirected |

**Remarks**

You should use Accept or Reject as soon as possible in the event handler invoked to handle an IncomingCall event. Doing so ensures the protocol layer in use will not timeout while attempting to connect to the Application Runtime Environment.

If the call is accepted, it may then be answered using the AnswerCall action. This does not have to occur immediately.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

## Metreos.CallControl.RemoteHangup Event

Indicates that the call has been terminated by the remote party.

**Type**

Non-triggering

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | Token used to identify this call |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

## Metreos.CallControl.StartTx Event

Indicates that an outbound media channel has been established.

**Type**

Non-triggering

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | Token used to identify this call |
| MmsId | System.String | ID of the media server handling media for this call |
| ConnectionId | System.String | ID of the media server connection handling media for this call |
| MediaTxIP | System.String | The IP address that media is being sent to |
| MediaTxPort | System.UInt32 | The port that media is being sent to |
| MediaTxCodec | System.String | The codec used on the send channel |
| MediaTxFramesize | System.UInt32 | The framesize used on the send channel |

**Remarks**

This should always occur in response to a MakeCall or AnswerCall action.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

## Metreos.CallControl.StopTx Event

Indicates that an outbound media channel has been closed.

**Type**

Non-triggering

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | Token used to identify this call |
| MmsId | System.String | ID of the media server handling media for this call |
| ConnectionId | System.String | ID of the media server connection handling media for this call |

**Remarks**

This event can occur if the remote party presses 'hold', for instance.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

## Metreos.CallControl.StartRx Event

Indicates that an inbound media channel has been established.

**Type**

Non-triggering

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | Token used to identify this call |
| MmsId | System.String | ID of the media server handling media for this call |
| ConnectionId | System.String | ID of the media server connection handling media for this call |
| MediaRxIP | System.String | The IP address that media is being received on |
| MediaRxPort | System.UInt32 | The port that media is being received on |
| MediaRxCodec | System.String | The codec used on the receive channel |
| MediaRxFramesize | System.UInt32 | The framesize used on the receive channel |

**Remarks**

This may, but is not guaranteed to, occur in response to a MakeCall or AnswerCall action.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

## Metreos.CallControl.MakeCall_Complete Event

Indicates that the requested call setup has completed successfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | Token used to identify this call |
| MmsId | System.String | ID of the media server handling media for this call |
| To | System.String | The final dialed number for this call |
| OriginalTo | System.String | The originally dialed number for this call |
| ConnectionId | System.String | ID of the connection to the media server handling this media for this call |
| ConferenceId | System.String | ID of the conference on the media server that this connection was placed in. Valid only if conference was specified as true. |
| MediaTxIP | System.String | The IP address that media is being sent to |
| MediaTxPort | System.UInt32 | The port that media is being sent to |
| MediaTxCodec | System.String | The codec used on the send channel |
| MediaTxFramesize | System.UInt32 | The framesize used on the send channel |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

### Metreos.CallControl.MakeCall_Failed Event

Indicates that the make call action did not succeed.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | Token used to identify this call |
| To | System.String | The final dialed number for this call |
| Reason | System.String | Text description of the reason for the call failure |
| ReasonCode | System.UInt32 | The numeric code for the reason for the call failure |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

Metreos JTAPI Provider version 2.0

## 2.2. Media Control

### 2.2.1. Actions

### Metreos.MediaControl.ReserveConnection Action

Reserves a connection on the specified media server or any of the media servers in the configured media resource group for the current application partition.

**Operation**

Synchronous

**Required Parameters**

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| MmsId | System.String | ID of the media server handling media for |

| | | |
|---|---|---|
| | | this call |
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |
| MmsId | System.UInt32 | ID of the media server handling media for this call |
| MediaRxControlPort | System.UInt32 | The RTCP port of the local media server |
| ConnectionId | System.String | Token used to identify this connection to the media server |
| MediaRxPort | System.UInt32 | The port at which the local media server wishes to receive media |
| MediaRxControlIP | System.String | The RTCP IP address of the local media server |
| MediaRxIP | System.String | The IP address at which the local media server wishes to receive media |

**Return Value**

Returns `Success` if the connection could be reserved; otherwise, `Failure` is returned.

**Remarks**

ReserveConnection is usually used in conjunction with CreateConnection. First the connection is reserved by using ReserveConnection, but the transmit properties of the connection have not been set. Using CreateConnection, you can then specify the transmit properties, such as `MediaTxIP` and MediaTxPort.

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.CreateConnection Action

Establishes an end-to-end connection to the media server using the supplied remote media address.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| MediaTxIP | System.String | The media IP address of the remote endpoint |
| MediaTxPort | System.Int32 | The media port of the remote endpoint |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | The ID of the call associated with this connection (hairpin only) |
| ConnectionId | System.String | Token used to identify this connection to the media server |
| MmsId | System.UInt32 | ID of the media server handling media for this call |
| MediaTxControlIP | System.String | The RTCP IP address of the remote endpoint |
| MediaTxControlPort | System.UInt32 | The RTCP port of the remote endpoint |
| MediaRxCodec | System.String | Incoming media encoding type. Valid values are G711u, G711a, G723, G729 |
| MediaRxFramesize | System.UInt32 | Incoming media framesize (ms) |
| MediaTxCodec | System.String | Incoming media encoding type. Valid values are G711u, G711a, G723, G729 |
| MediaTxFramesize | System.UInt32 | Outbound media framesize (ms) |
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |
| MmsId | System.UInt32 | ID of the media server handling media for this call |
| MediaRxControlPort | System.UInt32 | The RTCP port of the local media server |
| ConnectionId | System.String | Token used to identify this connection to the media server |
| MediaRxPort | System.UInt32 | The port at which the local media server wishes to receive media |

| | | |
|---|---|---|
| MediaRxControlIP | System.String | The RTCP IP address of the local media server |
| MediaRxIP | System.String | The IP address at which the local media server wishes to receive media |

**Return Value**

Returns `Success` if the connection could be fully connected; otherwise, `Failure` is returned.

**Remarks**

This action can either take a reserved connection and fully connect it (Tx and Rx properties are now associated with the connection), or create a completely new connection. The latter possibility removes the need to use ReserveConnection. `MmsId` can be used the same way it would be used in a ReserveConnection action.

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.ModifyConnection Action

Modifies an existing connection to the media server

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| MediaTxIP | System.String | The media IP address of the remote endpoint |
| MediaTxPort | System.Int32 | The media port of the remote endpoint |
| ConnectionId | System.String | Token used to identify this connection to the media server |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CallId | System.String | The ID of the call associated with this connection (hairpin only) |
| MmsId | System.UInt32 | ID of the media server handling media for this call |
| MediaTxControlIP | System.String | The RTCP IP address of the remote endpoint |
| MediaTxControlPort | System.UInt32 | The RTCP port of the remote endpoint |
| MediaRxCodec | System.String | Incoming media encoding type. Valid values are G711u, G711a, G723, G729 |
| MediaRxFramesize | System.UInt32 | Incoming media framesize (ms) |
| MediaTxCodec | System.String | Incoming media encoding type. Valid values are G711u, G711a, G723, G729 |
| MediaTxFramesize | System.UInt32 | Outbound media framesize (ms) |
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |
| MmsId | System.UInt32 | ID of the media server handling media for this call |
| MediaRxControlPort | System.UInt32 | The RTCP port of the local media server |
| ConnectionId | System.String | Token used to identify this connection to the media server |
| MediaRxPort | System.UInt32 | The port at which the local media server wishes to receive media |

| | | |
|---|---|---|
| MediaRxControlIP | System.String | The RTCP IP address of the local media server |
| MediaRxIP | System.String | The IP address at which the local media server wishes to receive media |

**Return Value**

Returns `Success` if the connection could be modified; otherwise, `Failure` is returned.

**Remarks**

ModifyConnection is used once a connection has been established. It may be necessary to modify some of the receive or transmit parameters of that connection at a later time.

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.DeleteConnection Action

Deletes the specified connection or conference.

**Operation**

Synchronous

**Required Parameters**

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ConferenceId | System.String | ID of the conference on the media server that this connection was placed in. Valid only if Conference was specified as true. |
| ConnectionId | System.String | Token used to identify this connection to the media server |
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |

**Return Value**

Returns `Success` if the connection could be deleted; otherwise, `Failure` is returned.

**Remarks**

Either `ConferenceId` or `ConnectionId` must be specified

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.Mute Action

Mutes the specified connection within a conference.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ConferenceId | System.String | ID of the conference on the media server that this connection was placed in. Valid only if Conference was specified as true. |
| ConnectionId | System.String | Token used to identify this connection to the media server |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |

**Return Value**

Returns `Success` if the connection could be muted; otherwise, `Failure` is returned.

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.Unmute Action

Unmutes the specified connection within a conference.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ConferenceId | System.String | ID of the conference on the media server that this connection was placed in. Valid only if Conference was specified as true. |
| ConnectionId | System.String | Token used to identify this connection to the media server |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |

**Return Value**

Returns `Success` if the connection could be unmuted; otherwise, `Failure` is returned.

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.Play Action

Initiates playback of the specified media file (with .wav or .vox extension) or speaks the text to the specified connection or conference

**Operation**

Asynchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Prompt1 | System.String | Name of a media file to play, with file extension |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Prompt2 | System.String | Name of a media file to play, with file extension |
| Prompt3 | System.String | Name of a media file to play, with file extension |
| ConnectionId | System.String | Token used to identify this connection to the media server |
| AudioFileSampleRate | System.UInt32 | Sample rate of the audio file (in kHz) |
| AudioFileEncoding | System.String | Encoding of the audio file: 'ulaw' or 'alaw' |
| State | System.String | Optional user state information to be returned when asynchronous command completes |
| CommandTimeout | System.UInt32 | Maximum time in which the media operation should complete |
| TermCondSilence | System.UInt32 | Amount of silence in milliseconds to observe before terminating the play action |
| TermCondNonSilence | System.UInt32 | Amount of non-silence in milliseconds to observe before terminating the media operation |
| TermCondMaxTime | System.UInt32 | Amount of time in milliseconds to wait before terminating the media operation |
| TermCondDigit | System.String | Digit on which to terminate the media operation |
| TermCondDigitList | System.String | Digit list to observe before terminating the media operation |
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |

**Return Value**

Returns `Success` if the connection could be muted; otherwise, `Failure` is returned.

**Remarks**

Up to three prompts may be specified to play in succession. If a termination condition is encountered, all prompts are cancelled.

Either a `ConnectionId` or a `ConferenceId` must be specified, but not both.

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.Record Action

Records audio from a connection or a conference.

**Operation**

Asynchronous

**Required Parameters**

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Filename | System.String | Name of the media file to create, with desired extension, '.vox' or '.wav' |
| Expires | System.UInt32 | Number of days until the file is deleted from the server. Defaults to one day. |
| ConnectionId | System.String | Token used to identify this connection to the media server |
| AudioFileSampleRate | System.UInt32 | Sample rate of the audio file (in kHz) |
| AudioFileEncoding | System.String | Encoding of the audio file: 'ulaw' or 'alaw' |
| State | System.String | Optional user state information to be returned when asynchronous command completes |
| CommandTimeout | System.UInt32 | Maximum time in which the media operation should complete |
| TermCondSilence | System.UInt32 | Amount of silence in milliseconds to observe before terminating the play action |
| TermCondNonSilence | System.UInt32 | Amount of non-silence in milliseconds to observe before terminating the media operation |
| TermCondMaxTime | System.UInt32 | Amount of time in milliseconds to wait before terminating the media operation |
| TermCondDigit | System.String | Digit on which to terminate the media operation |
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |

**Return Value**

| |
|---|
| Returns `Success` if the connection could be muted; otherwise, `Failure` is returned. |
| **Remarks** |
| If filename is not specified, the media server will create a random file name, which is returned in the Record_Complete event.<br>Either a `ConnectionId` or a `ConferenceId` must be specified, but not both. |
| **Requirements** |
| Metreos Framework version 2.0<br>Metreos Media Provider version 2.0 |

## Metreos.MediaControl.StopMediaOperation Action

Stops any currently executing media server operation on the specified connection.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ConnectionId | System.String | Token used to identify this connection to the media server |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |

**Return Value**

Returns `Success` if the connection could be unmuted; otherwise, `Failure` is returned.

**Remarks**

If an asynchronous event is terminated by this action, the terminationCondition returned by the event will have the value ‘userstop’.

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.DetectSilence Action

Monitor a connection for silence.

**Operation**

Asynchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ConnectionId | System.String | Token used to identify this connection to the media server |
| SilenceTime | System.UInt32 | Amount of silence to observe (ms) |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |
| CommandTimeout | System.UInt32 | Maximum time in which the media operation should complete |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |

**Return Value**

Returns `Success` if the connection could be unmuted; otherwise, `Failure` is returned.

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.CreateConference Action

Creates a conference and places the specified connection into it.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ConnectionId | System.String | Token used to identify this connection to the media server |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| SoundToneOnJoin | System.Boolean | Indicates whether tones are played when participants join the conference |
| CallId | System.String | The ID of the call associated with this connection (hairpin only) |
| MediaTxIP | System.String | The IP address that media is being sent to |
| MediaTxPort | System.UInt32 | The port that media is being sent to |
| MediaTxCodec | System.String | The codec used on the send channel |
| MediaTxFramesize | System.UInt32 | The framesize used on the send channel |
| MediaTxControlIP | System.String | The RTCP IP address of the remote endpoint |
| MediaTxControlPort | System.UInt32 | The RTCP port of the remote endpoint |
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |
| ConnectionId | System.String | Token used to identify this connection to the media server |
| ConferenceId | System.String | Token used to identify the newly created conference |

**Return Value**

Returns `Success` if the connection could be unmuted; otherwise, `Failure` is returned.

**Remarks**

Creates a conference and places one connection into that conference. A conference can not be created with zero connections.

That connection can already be fully connected (Tx and Rx conditions both established using ReserveConnection and/or CreateConnection), or merely created through the ReserveConnection action, in which case the `MediaTxIP`, `MediaTxPort` must also be specified.

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.JoinConference Action

Moves the specified connection into a conference.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ConnectionId | System.String | Token used to identify this connection to the media server |
| ConferenceId | System.String | ID of the conference on the media server that this connection was placed in. |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |

**Return Value**

Returns `Success` if the connection could be unmuted; otherwise, `Failure` is returned.

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.LeaveConference Action

Removes the specified connection from a conference.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ConnectionId | System.String | Token used to identify this connection to the media server |
| ConferenceId | System.String | ID of the conference on the media server that this connection was placed in. |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |

**Return Value**

Returns `Success` if the connection could be unmuted; otherwise, `Failure` is returned.

**Remarks**

If the connection being removed from the conference is the last connection in that conference, then the conference is automatically destroyed.

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.GatherDigits Action

Instructs the media server to watch for a particular set of DTMF digits on the specified connection.

**Operation**

Asynchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ConnectionId | System.String | Token used to identify this connection to the media server |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| FlushBuffer | System.Boolean | Indicates that any digits gathered previously should be cleared |
| TermCondMaxTime | System.UInt32 | Amount of time in milliseconds to wait before terminating the media operation |
| TermCondDigit | System.String | Digit on which to terminate the media operation |
| TermCondDigitList | System.String | Digit list to observer before terminating the media operation |
| TermCondDigitPattern | System.String | A specific digit pattern to observe before terminating the media operation |
| TermCondInterDigitDelay | System.Uint32 | The maximum amount of time between digits to allow before terminating the media operation |
| TermCondMaxDigits | System.Uint32 | Number of digits to receive before terminating the media operation |
| State | System.String | Optional user state information to be returned when asynchronous command completes |
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |

**Return Value**

Returns `Success` if the connection could be unmuted; otherwise, `Failure` is returned.

**Remarks**

| |
|---|
| The media server will use any digits previously entered in the digit buffer for the connection unless the `FlushBuffer` parameter is set to `true`. The digit buffer is flushed automatically after this action has fully completed (i.e., the asynchronous callback is fired). |
| **Requirements** |
| Metreos Framework version 2.0<br>Metreos Media Provider version 2.0 |

## Metreos.MediaControl.SendDigits Action

Inserts DTMF digits into the digit buffer for the specified connection.

**Operation**

Synchronous

**Required Parameters**

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time to wait before the Application Runtime Environment should force the action to fail |
| ConnectionId | System.String | |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |

**Return Value**

Returns `Success` if the connection could be unmuted; otherwise, `Failure` is returned.

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.PlayTone Action

Plays a tone to the specified connection or conference.

**Operation**

Asynchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ConnectionId | System.String | Token used to identify this connection to the media server |
| Digits | System.String | The digit(s) to send |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ConnectionId | System.String | Token used to identify this connection to the media server |
| ConferenceId | System.String | Token used to identify a conference |
| Frequency1 | System.UInt32 | Frequency to be played (Hz). Valid range: 200 - <3000 |
| Frequency2 | System.UInt32 | Frequency to combine with the first frequency (Hz). Valid range: 200 - <3000 |
| Amplitude1 | System.Int32 | Amplitude of tone (dB). Valid range: -40 - <0 |
| Amplitude2 | System.Int32 | Amplitude to combine with first amplitude (dB). Valid range: -40 - <0 |
| Duration | System.UInt32 | Length of time to play tone (ms) |
| State | System.String | Optional user state information to be returned when asynchronous command completes |
| TermCondSilence | System.UInt32 | Interval of silence in milliseconds to observe before terminating the play action |
| TermCondNonSilence | System.UInt32 | Interval of non-silence in milliseconds to observe before terminating the media operation |
| TermCondMaxTime | System.UInt32 | Interval in milliseconds to wait before terminating the media operation |
| TermCondDigit | System.String | Digit on which to terminate the media operation |
| Timeout | System.Int32 | Interval to wait before the Application Runtime Environment should force the action to fail |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |

**Return Value**

Returns `Success` if the connection could be unmuted; otherwise, `Failure` is returned.

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

### 2.2.2. Events

**Metreos.MediaControl.Play_Complete Event**

Indicates that the requested play has completed successfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| TerminationCondition | System.String | Describes the condition which caused the media operation to stop |
| State | System.String | Optional user state information to be returned when asynchronous command completes |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.Play_Failed Event

Indicates that the requested play did not complete successfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |
| State | System.String | Optional user state information to be returned when asynchronous command completes |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.Record_Complete Event

Indicates that the requested record has completed successfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| TerminationCondition | System.String | Describes the condition which caused the media operation to stop |
| Filename | System.String | Name of the media file created |
| State | System.String | Optional user state information to be returned when asynchronous command completes |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.Record_Failed Event

Indicates that the requested record did not complete successfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |
| State | System.String | Optional user state information to be returned when asynchronous command completes |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.GatherDigits_Complete Event

Indicates that the requested gather digits has completed successfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| TerminationCondition | System.String | Describes the condition which caused the media operation to stop |
| Digits | System.String | The digits gathered, including those which contributed to the termination condition |
| State | System.String | Optional user state information to be returned when asynchronous command completes |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.GatherDigits_Failed Event

Indicates that the requested gather digits did not complete successfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |
| State | System.String | Optional user state information to be returned when asynchronous command completes |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.DetectSilence_Complete Event

Indicates that the requested detect silence has completed successfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| TerminationCondition | System.String | Describes the condition which caused the media operation to stop |
| Digits | System.String | The digits gathered, including those which contributed to the termination condition |
| State | System.String | Optional user state information to be returned when asynchronous command completes |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.DetectSilence_Failed Event

Indicates that the requested detect silence did not complete successfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |
| State | System.String | Optional user state information to be returned when asynchronous command completes |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.PlayTone_Complete Event

Indicates that the requested play tone has completed successfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| TerminationCondition | System.String | Describes the condition which caused the media operation to stop |
| Digits | System.String | The digits gathered, including those which contributed to the termination condition |
| State | System.String | Optional user state information to be returned when asynchronous command completes |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## Metreos.MediaControl.PlayTone_Failed Event

Indicates that the requested detect silence did not complete successfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultCode | System.String | Describes the nature of the media server error, if any |
| State | System.String | Optional user state information to be returned when asynchronous command completes |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Provider version 2.0

## 2.3. HTTP

The HTTP provider supports the concept of sessions. A session is created the first time a script instance uses a **`Metreos.Providers.Http.SendResponse`** action. The HTTP Provider will then proxy any session-bound HTTP request to the script instance creating the session. A request to the HTTP Provider can signal that it is communicating to a session through one of three mechanisms:

- A **metreosSessionID** query parameter in the URL
- A header in the incoming request with a header named **`Metreos-SessionID`**, containing the Session GUID as the value
- A cookie with name **`metreosSessionId`**, and value of the Session GUID

The client making these requests will be able to determine the Session GUID through one of two mechanisms:

- A header named **`Metreos-SessionID`** containing the Session GUID as a value is present in all 200 OK responses from the Application Runtime Environment. An intelligent client can then save and use Session GUID value in subsequent requests.
- For support of most browser-type clients, a **Set-Cookie** header will be sent in a response to the client in a properly formed **`Metreos.Providers.Http.SendResponse`**. The Cookie will have the name **`metreosSessionId`**, and the value will be the Session GUID.

The session GUID is the same value as the routing GUID of the script instance.

You should supply a **Set-Cookie** header in a **`SendResponse`** action in HTTP requests. The **Set-Cookie** header is necessary so the request can pass the cookie in all subsequent requests. The request can then gain access to any non-triggering **`Metreos.Provider.Http.GotRequest`** event handlers contained in this script.

Sessions fire an event after a configurable amount of time after initialization to the script instance. You can then handle The **`SessionExpired`**, can then be handled in whatever way is required.

### 2.3.1. Actions

## Metreos.Providers.Http.SendResponse Action

Send an HTTP response to a request that was previously received.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| remoteHost | System.String | Address of host who sent the original request |
| responseCode | System.Int32 | Numeric code of HTTP response |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| body | System.String | Body of the response |
| responsePhrase | System.String | Description of response |
| Content-Type | System.String | The type of content being sent. |

**Result Data Fields**

None

**Return Value**

Returns "`success`" if the response was sent; otherwise, "`failure`" is returned.

**Remarks**

When sending responses to HTTP requests the application must specify **`responseCode`** and **`remoteHost`**. The **`remoteHost`** parameter is used by the HTTP provider to correlate the response and the originating HTTP request. Set the value of the **`remoteHost`** parameter equal to the value of the parameter of the same name received in the **`GotRequest`** event.

Optionally, an application may specify any other headers as parameters in the **`SendResponse`** action. An application could, for example, specify to the client how to cache the response. To do so, a parameter named **`Cache-Control`** (HTTP specific header field) must be inserted in the action with an appropriate value, such as **`no-cache`**.

**Requirements**

Metreos Framework version 2.0

Metreos HTTP Provider version 2.0

### 2.3.2. Events

## Metreos.Providers.Http.GotRequest Event

Event fired when an HTTP request is received by the Application Runtime Environment.

**Type**

Hybrid

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| host | System.String | Host portion of the request URI (may include port info) |
| hostname | System.String | Host portion of the request URI (no port info) |
| method | System.String | HTTP method (POST, GET, etc) |
| port | System.Int32 | Port portion of the request URI |
| query | System.String | Query string portion of the request URI |
| remoteHost | System.String | The IP address and port of the remote client |
| remoteIpAddress | System.String | The IP address of the remote client |
| url | System.String | Path portion of the request URI |
| [headers] | System.String | Any number of headers in the HTTP request |

**Remarks**

This event will fire to the Application Runtime Environment when either a **GET** or **POST** request arrives at the Application Runtime Environment. All HTTP headers will be maintained and can be accessed from the event parameters by using the header name as the parameter name.

**Requirements**

Metreos Framework version 2.0

Metreos HTTP Provider version 2.0

## Metreos.Providers.Http.SessionExpired Event

Event fired when a HTTP request is received by the Application Runtime Environment.

**Type**

Non-triggering

**Event Parameters**

**Remarks**

The HTTP Provider regularly checks all outstanding sessions to determine if any have expired. If a session is expired, the **Metreos.Providers.Http.SessionExpired** event is sent to the script instance that was started when the session was created.

**Requirements**

Metreos Framework version 2.0

Metreos HTTP Provider version 2.0

### 2.3.3. Types

## Metreos.Types.Http.QueryParamCollection Type

Parses the query parameter section of a HTTP URI, populating a collection with key-value pairs. The collection can then be used to determine which query parameters came in with an HTTP request.

**Input Types**

| Type Name | Description |
|---|---|
| string | The section of the URI containing the query parameters, exclusive or inclusive of the beginning '?'. |

**Accessible Public Methods**

| Method Name | Description |
|---|---|
| string this[string name] | Returns the value of the named query parameter. An empty string is a valid value for a query parameter. If a query parameter was not defined in the supplied query parameter URI fragment to pars, it will return null. |
| string this[int index] | Returns the value of the named query parameter. An empty string is a valid value for a query parameter. If a query parameter was not defined in the supplied query parameter URI fragment to parse, it will return null. |
| string GetNameAt( int index ) | Returns the name of the query parameter at the given index. |
| void Clear() | Clears the collection. |

**Accessible Public Properties**

| Property Name | Description |
|---|---|
| int Count | Returns the number of query parameters in the collection. |

**Remarks**

The most commonly intended use of this type is to initialize it with the **`query`** event parameter accompanying **Metreos.Providers.Http.GotRequest**, after which it can be used.

**Requirements**

Metreos Framework version 2.0

## 2.4. Application Control

### 2.4.1. Actions

#### Metreos.ApplicationControl.SetSessionData Action

Sets the value of a session data variable.

**Operation**

Synchronous

**Required Parameters**

None.

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| [parameters] | System.Object | Parameters that will be stored in the session data Hashtable. |

**Result Data Fields**

None.

**Return Value**

**`success`** if the modification was successful, otherwise `failure`

**Remarks**

Any number of **SessionData** entries can be specified. The name of a defined parameter will be used as a key in the **`sessionData.CustomData System.Collections.Hashtable`**, and the value specified for the defined parameter will be used as the value for the previously mentioned key. If, for example, a parameter **`a`** is created, and the literal string **`true`** is assigned to it, **`true`** will be stored in the **`sessionData.CustomData System.Collections.Hashtable. a`** is the key, and **`true`** is the value. In order to access session data in a custom code action, **`SessionData`** must be specified as a method parameter. The value is stored as an object inside the **`Hashtable`**. In most case you will be required to cast any retrieved elements to the proper type before using them.

**Requirements**

Metreos Framework version 2.0

## Metreos.ApplicationControl.EnableScript Action

Enables a slave script for execution.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ScriptName | System.String | Name of the script that you wish to enable. |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| [parameters] | System.Object | Triggering parameters for the script that you wish to enable. |

**Result Data Fields**

None.

**Return Value**

`success` if the operation was successful, otherwise `failure`

**Remarks**

This action allows you to enable a slave script from a running application. You must specify the name of the script that you want to start and all the required triggering parameters for that script. The names of the parameters you specify in this action must match the names of the triggering parameters of the target script.

**Requirements**

Metreos Framework version 2.0

## Metreos.ApplicationControl.CallFunction Action

Calls a function inside the currently executing application.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| FunctionName | System.String | The name of the function to invoke |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| [parameters] | System.String | Any number of parameters that can be passed in to the invoked function. |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| [resultDataFields] | System.String | Any number of result data fields passed back from the exited invoked function |

**Return Value**

The return value of this action is specified by the function that is being called.

**Remarks**

Well-designed applications often have repeated logic that can be re-factored into a single function. This action allows applications to define stand-alone, non-event handling functions and to invoke those functions from their application logic. However, this action can also be used to call event-handling functions provided the `CallFunction` action contains all the required parameters expected by the event handler.

**Requirements**

Metreos Framework version 2.0

## Metreos.ApplicationControl.EndScript Action

Exit and cleanup the currently executing application instance.

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

None

**Result Data Fields**

None

**Return Value**

None.

**Remarks**

Application instances will continue to run until an **`EndScript`** action is processed. This action informs the Application Runtime Environment that the currently executing application instance is ready to be terminated. After processing an **`EndScript`** action no further events will be received by the application instance and no further actions may be invoked by the application instance.

**Requirements**

Metreos Framework version 2.0

## Metreos.ApplicationControl.EndFunction Action

Exits the currently executing function.

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ReturnValue | System.String | The value used as the branching condition of the CallFunction which invoked the current function |
| [resultDataFields] | System.String | Any number of result data fields returned to the CallFunction which invoked the function |

**Result Data Fields**

None

**Return Value**

None.

**Remarks**

Functions may exit at any time. If a function for an event handler exits, then the application instance will continue to run waiting for further events. When a function exits after being called by another function, execution of the application returns to the calling function.

Functions may return any number of result data fields to the calling function by adding parameters to the **`EndFunction`** action. Any parameters present in the action will be passed to the calling function for processing.

Return values may be specified in the **`ReturnValue`** parameter. The return value allows the calling function to branch on the success or failure of the function that was called.

**Requirements**

Metreos Framework version 2.0

## Metreos.ApplicationControl.Forward Action

Forwards all subsequent events for a particular application script type to another application script type.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ToGuid | System.String | The routing GUID of the running application in which to send the event |

**Optional Parameters**

None

**Result Data Fields**

None

**Return Value**

None.

**Remarks**

This action instructs the Application Runtime Environment to forward all future events and responses to another application. It requires a **`ToGuid`** parameter, which holds the routing GUID of the target application. In order to get that information, the target application would have to make it available through the registry, a database, or some other thread-safe mechanism. Once this action is executed, no further actions are taken and the application instance is reset.

**Requirements**

Metreos Framework version 2.0

## Metreos.ApplicationControl.SendEvent Action

Fires an event to another application.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| EventName | System.String | The name for the event |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ToGuid | System.String | The routing GUID of the running application in which to send the event |
| [parameters] | System.String | Any number of parameters to pass into the event |

**Result Data Fields**

None

**Return Value**

Returns `success` if the event was fired; otherwise, `failure` is returned.

**Remarks**

This action enables application scripts to communicate with other application scripts installed on the Application Runtime Environment. The **`EventName`** parameter specifies the name of the event that will be fired. The **`ToGuid`** parameter specifies the routing GUID of a running application script. If the **`ToGuid`** parameter is not specified, the event will be handled like any other triggering event.

Applications may specify any number of parameters in the **`SendEvent`** action. These parameters are preserved and passed to the application instance receiving the event.

**Requirements**

Metreos Framework version 2.0

## Metreos.ApplicationControl.Sleep Action

Causes execution of the script to pause for the requested amount of time.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| SleepTime | System.Int32 | The amount of time to sleep in milliseconds. |

**Optional Parameters**

None

**Result Data Fields**

None

**Return Value**

Returns **`success`** if the event was fired; otherwise, **`failure`** is returned.

**Remarks**

No other events can be handled while the script is sleeping.

**Requirements**

Metreos Framework version 2.0

## Metreos.ApplicationControl.Assign Action

Assigns up to 4 action parameters to result data fields.

**Operation**

Synchronous

**Required Parameters**

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Value | System.Object | A value to assign to ResultData |
| Value2 | System.Object | A value to assign to ResultData2 |
| Value3 | System.Object | A value to assign to ResultData3 |
| Value4 | System.Object | A value to assign to ResultData4 |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | System.Object | If used, Value must also be defined |
| ResultData2 | System.Object | If used, Value2 must also be defined |
| ResultData3 | System.Object | If used, Value3 must also be defined |
| ResultData4 | System.Object | If used, Value4 must also be defined |

**Return Value**

The Assign command can fail if any one of the assignments causes a type cast exception. In the case of an assignment failing, all assignments will not be made.

**Remarks**

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.Log.Write Action

Outputs a message to the log of the application.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| LogLevel | System.String | The trace level of the log message |
| Message | System.String | The log message |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| ApplicationName | System.String | The name to report as the source of the message |

**Result Data Fields**

None

**Return Value**

Returns `success` if the log message was output; otherwise, `failure` is returned.

**Remarks**

Applications may use this action to write informative log messages to the application log for debugging and other purposes. Log messages output by applications will be processed by the Application Runtime Environment's log subsystem and potentially output to multiple locations (console, file, Windows event log, etc).

The **`LogLevel`** parameter specifies the priority of the log message. It may be either `Verbose`, `Info`, `Warning`, or `Error`. This parameter is used by the Application Runtime Environment to effectively filter the log messages for various output devices.

**Requirements**

Metreos Framework version 2.0

## 2.5. Conditionals

This set of actions allows testing of and branching on user-defined conditions. The procedure for using these actions is to

1. Drag a conditional onto the canvas
2. Specify the boolean condition in one of the value fields
3. Specify connections to other nodes using branch conditions specific to each action.

The **`success`** and **`failure`** branch conditions do not apply to conditional actions.

### 2.5.1. Actions

#### Metreos.Native.Conditional.If Action

Returns `true` if the specified condition is true, returns `false` otherwise.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Value1 | System.Boolean | The value on which to perform the check. |

**Optional Parameters**

None

**Result Data Fields**

None

**Return Value**

Returns `true` or `false`, depending on what `Value1` evaluated to.

**Remarks**

This action takes a statement that evaluates either **`true`** or **`false`**. If the evaluated statement is **`true`**, the action returns **`true`** as its return value. If the evaluated statement is **`false`**, the action returns **`false`** In order to branch on this event, the branch conditions must be set to one of the following: **`true`**, **`false`**, or `default`.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.Conditional.Switch Action

Allows branching on a user defined object.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| SwitchOn | System.Object | The value on which to perform the switch. |

**Optional Parameters**

None

**Result Data Fields**

None

**Return Value**

Returns SwitchOn.ToString(), or `null` if SwitchOn is `null`.

**Remarks**

This action returns the value of the method **`ToString()`** on whatever object was passed as the **SwitchOn** argument. This allows the application developer to specify custom branch conditions. For example, if one passes a string variable into this action, one can specify branching conditions such as **`bob`** or **`tom`**. These branches will be taken if they match the value of **`SwitchOn.ToString()`**. Using a `default` branch is recommended in case none of the user-specified branches succeed.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.Conditional.Compare Action

Compares two values to each other.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Value1 | System.Object | The first value. |
| Value2 | System.Object | The second value. |

**Optional Parameters**

None

**Result Data Fields**

None

**Return Value**

Returns **`equal`** if the two values are equal to each other, and it returns **`unequal`** if they are not.

**Remarks**

None.

**Requirements**

Metreos Framework version 2.0

## 2.6. Database

The database actions allow the application to interface with an external database server such as Oracle or MySQL.

### 2.6.1. Actions

#### Metreos.Native.Database.OpenDatabase Action

Opens a connection to an Oracle, SqlServer, ODBC, MySQL.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| DSN | System.String | The DSN string for the desired database. |
| Name | System.String | Name of the connection. |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Type | System.String | The type of the database connection. |

**Result Data Fields**

None

**Return Value**

Returns `success` if the database connection was successful, returns `failure` if it was not.

**Remarks**

The **`DSN`** string specifies the information required by the action in order to find the database and connect to it. **`Name`** assigns a name to the connection, and can later be used by other database actions to refer to this specific connection. Valid values for Type are **oracle**, **sqlserver**, **odbc**, or **mysql**.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.Database.ExecuteQuery Action

Executes a query against a database connection.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Query | System.String | The query you want to execute on the database server. |
| Name | System.String | Name of the connection on which to perform the query. |

**Optional Parameters**

None

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultSet | System.Data.DataTable | The result of the query. |

**Return Value**

Returns `success` if the query executed successfully, returns `failure` if it did not.

**Remarks**

**`Name`** specifies the name of a connection that was earlier created using the **`OpenDatabase`** action. **`Query`** is a SQL statement that you want executed on the database. The values returned from the query may be placed into a variable of type **`System.Data.DataTable`**.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.Database.ExecuteCommand Action

Executes a command against a database connection.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Command | System.String | The command you want to execute on the database server. |
| Name | System.String | Name of the connection on which to perform the query. |

**Optional Parameters**

None

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| RowsAffected | System.Int32 | The number of rows affected by the command. |

**Return Value**

Returns **`success`** if the command executed successfully, returns **`failure`** if it did not.

**Remarks**

`Name` specifies the name of a connection that was earlier created using the **`OpenDatabase`** action. `Command` is a SQL statement that you want executed on the database.

**Requirements**

Metreos Framework version 2.0

## 2.7. Cisco IP Phone

This set of native actions provides utilities for the creation of complex displays for Cisco IP phones. The general procedure for creating a display is to

1. Declare a variable of the appropriate type
2. Execute the corresponding "Create..." action to provide details
3. Execute one or more "Add..." actions to add various components to the display such as soft keys and icons.

When the variable has been initialized and populated with all desired components, it may be sent in the body of a **200 OK** response to a request from the phone. The request must be accompanied with a text/xml **`Content-Type`**. The one exception to this rule is an Execute object. The Execute object is created in the same was as any other request, but is transported in the body of an HTTP **PUT** request using the **`SendExecute`** command.

### 2.7.1. Actions

## Metreos.Native.CiscoIpPhone.AddDirectoryEntry Action

Add an entry to a Cisco IP Phone directory.

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Name | System.String | Name of the directory entry |
| Telephone | System.String | Telephone number of the directory entry |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ReturnValue | System.String | Conforms to the XML representing a directory entry in a Metreos.Types.CiscoIpPhone.Directory type |

**Return Value**

Returns a properly formatted Cisco IP Phone directory entry. The **`returnValue`** parameter should be assigned to the **`Metreos.Types.CiscoIpPhone.Directory`** variable of the Cisco IP Phone Directory to be updated.

**Remarks**

When adding directory entries to a Cisco IP Phone directory the application should define a variable of type **`Metreos.Types.CiscoIpPhone.Directory`**. The application should then assign the **`returnValue`** parameter of the **`AddDirectoryEntry`** action to the variable that is holding the Cisco IP Phone directory. Doing so allows the directory object to insert the entry created by the **`AddDirectoryEntry`** action.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoIpPhone.AddIconItem Action

Adds an icon item to an icon menu.

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Data | System.String | Byte array of the data comprising the icon |
| Depth | System.UInt16 | The number of bits used to determine the colors available for the icon |
| Height | System.UInt16 | The height of the icon (pixels) |
| Index | System.UInt16 | The index of the icon. This value is referenced by the AddMenuItem action |
| Width | System.UInt16 | The width of the icon (pixels) |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | CiscoIpPhone. CiscoIPPhoneInput ItemType | Conforms to the XML representing an icon item in a Metreos.Types.CiscoIpPhone.IconMenu type |

**Return Value**

Returns a properly formatted Cisco IP Phone icon item entry.

**Remarks**

Icons may be added to Cisco IP Phone icon menus. It is unnecessary to add the icon more than once and the icon is identified by its **`Index`** parameter. You can reference the icon later when adding menu items by referring to the value of the **`Index`** parameter.

The **Data** parameter of the **`AddIconItem`** action must be a properly formatted CIP image. Refer to the *Cisco IP Phone SDK* for more information about the CIP format.

When building a Cisco IP Phone icon menu, the application should define a variable of type **`Metreos.Types.CiscoIpPhone.IconMenu`**. The application should then assign the **`ResultData`** parameter of the **`AddIconItem`** action to the variable holding the Cisco IP Phone icon menu. Doing this allows the icon menu object to insert the entry created by the **`AddIconItem`** action.

**Requirements**

Metreos Framework version 2.0

# Metreos.Native.CiscoIpPhone.AddInputItem Action

Adds an input field to an input object.

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| DefaultValue | System.String | The value to initially display for the input item |
| DisplayName | System.String | The name of the input |
| InputFlags | System.String | Indicates the behavior of the input item |
| QueryStringParam | System.String | The name of the query parameter that will be sent in the HTTP GET executed by a Cisco IP Phone, if the input item is selected |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | Metreos.Types.CiscoIpPhone.CiscoIPPhoneInputItemType | Conforms to the XML representing an input item in a Metreos.Types.CiscoIpPhone.Input type |

**Return Value**

Returns a properly formatted Cisco IP Phone input item. The **`ResultData`** parameter should be assigned to **`Metreos.Types.CiscoIpPhone.Input`** for the input item you want to add.

**Remarks**

The **`InputFlags`** parameter specifies the type of input field to generate. Valid input flags are

- A – ASCII text. Permissible values are upper /lowercase alphanumeric and special characters.
- T – Telephony number. Permissible values are numbers, pound (**#**), and asterisk (*****).
- N – Numeric. Permissible values are numbers.
- E – Equation. Permissible values are numbers and the special math symbols.
- U – Uppercase. Permissible values are uppercase letters.
- L – Lowercase. Permissible values are lowercase letters.
- P – Password. Individual characters display as they are entered and then change to an asterisk to mask the entered value.

When building a Cisco IP Phone input object, the application should define a variable of type **`Metreos.Types.CiscoIpPhone.Input`**. The application should then assign the **`ResultData`** parameter of the **`AddInputItem`** action to the variable that is holding the Cisco IP Phone input object. Doing so allows the input object to insert the entry created by the **`AddInputItem`** action.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoIpPhone.AddMenuItem Action

Adds a menu item to a menu object.

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| IconIndex | System.UInt16 | If this action is appending to a **Metreos.Types.CiscoIpPhone.IconMenu**, this value indicates which index to reference in displaying an icon |
| Name | System.String | Name of the menu item |
| TouchArea-X1 | System.Int16 | Represents the X-coordinate of the upper-left corner of a bounded area that will cause this menu item to be selected when touched (pixels) *touch screen enabled phones only* |
| TouchArea-X2 | System.Int16 | Represents the X-coordinate of the lower-right corner of a bounded area that will cause this menu item to be selected when touched (pixels). For touch screen enabled phones only. |
| TouchArea-Y1 | System.Int16 | Represents the Y-coordinate of the upper-left corner of a bounded area that will cause this menu item to be selected when touched (pixels). For touch screen enabled phones only. |
| TouchArea-Y2 | System.Int16 | Represents the Y-coordinate of the lower-right corner of a bounded area that will cause this menu item to be selected when touched (pixels). For touch screen enabled phones only. |
| URL | System.String | The URL to request if this particular menu item is selected |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | Metreos.Types.CiscoIpPhone.CiscoIPPhoneMenuItemType | Conforms to the XML representing a menu item in a Metreos.Types.CiscoIpPhone.Menu, Metreos.Types.CiscoIpPhone.IconMenu, Metreos.Types.CiscoIpPhone.GraphicMenu, or a Metreos.Types.CiscoIpPhone.GraphicFileMenu type |

**Return Value**

Returns a properly formatted Cisco IP Phone menu item entry. The **`ResultData`** parameter

| |
|---|
| should be assigned to any menu type variable (listed above) that the application wishes to add the input item to. |
| **Remarks** |
| The parameters of this action indicate display name, icon to use (if appropriate), link address, and placement of the menu item on the phone display. For touch screen enabled phones, a touch area can be designated for the item as well. |
| **Requirements** |
| Metreos Framework version 2.0 |

## Metreos.Native.CiscoIpPhone.AddSoftKeyItem Action

Adds a soft key item to almost any Cisco IP phone format type.

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Name | System.String | Name of the soft key item |
| Position | System.UInt16 | The position of the soft key item |
| URL | System.String | The URL to request if this particular soft key item is selected |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | Metreos.Types.CiscoIpPhone.CiscoIPPhoneSoftKeyType | Conforms to the XML representing a soft key item in a Metreos.Types.CiscoIpPhone.Text, Metreos.Types.CiscoIpPhone.Directory, Metreos.Types.CiscoIpPhone.Menu, Metreos.Types.CiscoIpPhone.Input, Metreos.Types.CiscoIpPhone.IconMenu, Metreos.Types.CiscoIpPhone.Image, Metreos.Types.CiscoIpPhone.ImageFile, Metreos.Types.CiscoIpPhone.GraphicMenu, or a Metreos.Types.CiscoIpPhone.GraphicFileMenu type |

**Return Value**

Returns a properly formatted Cisco IP Phone soft key. The **`ResultData`** parameter should be assigned to almost any variable type for the soft key you want to add.

**Remarks**

This action adds a soft key to any screen on the Cisco IP phone. Soft keys appear at the bottom of the screen and correspond to physical keys located immediately underneath the display. The **`URL`** parameter indicates where the phone retrieves display information when the user presses the defined soft key.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoIpPhone.CreateDirectory Action

Creates a directory display

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Prompt | System.String | Prompt to display on the Cisco IP phone |
| Title | System.String | Title to display on the Cisco IP phone |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | Metreos.Types.CiscoIpPhone.CiscoIPPhoneDirectoryType | Represents a minimal CiscoIpPhone.Directory message. This value can then be aggregated by a AddDirectoryEntry and a AddSoftKeyItem |

**Return Value**

Returns a properly formatted Cisco IP Phone directory display. The **`ResultData`** parameter should be assigned to a variable of type **`Metreos.Types.CiscoIpPhone.DirectoryType`**.

**Remarks**

This action creates a directory object and fills in only the basic properties. Individual directory items can be added by calling **`AddDirectoryEntry`** and specifying the resultant directory object of this operation as the target of the **`AddDirectoryEntry`** action.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoIpPhone.CreateExecute Action

Creates an "execute" command which can be pushed to the phone.

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| URL1 | System.String | Any valid URI |
| URL2 | System.String | Any valid URI |
| URL3 | System.String | Any valid URI |
| Priority1 | System.UInt16 | A valid priority |
| Priority2 | System.UInt16 | A valid priority |
| Priority3 | System.UInt16 | A valid priority |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | Metreos.Types.CiscoIpPhone.CiscoIPPhoneExecuteType | Represents a CiscoIpPhone.Execute message. This value cannot be further aggregated. |

**Return Value**

Returns a properly formatted Cisco IP Phone execute command. The **`ResultData`** parameter should be assigned to a variable of type **`Metreos.Types.CiscoIpPhone.Execute`**.

**Remarks**

This action creates a command which can be pushed to the phone by way of the **`SendExecute`** action. See the *Cisco IP Phone SDK* for more details on the valid URI schemes and priority codes supported by the Cisco IP Phones.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoIpPhone.CreateInput Action

Creates an input display.

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Prompt | System.String | Prompt to display on the Cisco IP phone |
| Title | System.String | Title to display on the Cisco IP phone |
| URL | System.String | The base URL used when an input item is selected |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | Metreos.Types.CiscoIpPhone.CiscoIPPhoneInputType | Represents a minimal CiscoIpPhone.Input message. This value can then be aggregated by a AddInputItem and a AddSoftKeyItem |

**Return Value**

Returns a properly formatted Cisco IP Phone input display. The **`ResultData`** parameter should be assigned to a variable of type **`Metreos.Types.CiscoIpPhone.Input`**.

**Remarks**

This action creates an input object and fills in only the basic properties. Individual input items can be added by calling **`AddInputItem`** and specifying the resultant input object of this operation as the target of the **`AddInputItem`** action.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoIpPhone.CreateIconMenu Action

Creates an icon menu display

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Prompt | System.String | Prompt to display on the Cisco IP phone |
| Title | System.String | Title to display on the Cisco IP phone |
| URL | System.String | URL that the icon menu points to |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | Metreos.Types.CiscoIpPhone.CiscoIPPhoneInputType | Represents a minimal CiscoIpPhone.IconMenu message. This value can then be aggregated by a AddMenuItem, AddIconItem, and a AddSoftKeyItem |

**Return Value**

Returns a properly formatted Cisco IP Phone icon menu display. The **`ResultData`** parameter should be assigned to a variable of type **`Metreos.Types.CiscoIpPhone.IconMenu`**.

**Remarks**

This action creates an icon menu object and fills in only the basic properties. Individual input items can be added by calling **`AddMenuItem`** or **`AddIconItem`** and specifying the resultant icon menu object of this operation as the target of the other actions.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoIpPhone.CreateImage Action

Creates an image display

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Data | System.String | Byte array of the data comprising the image |
| Depth | System.UInt16 | The number of bits used to determine the colors available for the image |
| Height | System.UInt16 | The height of the image (pixels) |
| LocationX | System.Int16 | The X-coordinate of the upper-left corner of the image to display (pixels) |
| LocationY | System.Int16 | The Y-coordinate of the upper-left corner of the image to display (pixels) |
| Prompt | System.String | Prompt to display on the Cisco IP phone |
| Title | System.String | Title to display on the Cisco IP phone |
| Width | System.UInt16 | The width of the image (pixels) |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | Metreos.Types.CiscoIpPhone.CiscoIPPhoneImageType | Represents a minimal CiscoIpPhone.Image message. This value can then be aggregated by a AddSoftKeyItem |

**Return Value**

Returns a properly formatted Cisco IP Phone image display. The **`ResultData`** parameter should be assigned to a variable of type **`Metreos.Types.CiscoIpPhone.Image`**.

**Remarks**

This action creates a properly formatted image object to be displayed on the Cisco IP Phone.

The `Data` parameter of the **`CreateImage`** action must be a properly formatted CIP image. Refer to the 'Cisco IP Phone SDK' for more information.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoIpPhone.CreateImageFile Action

Creates a file image display (models 7970 or greater only)

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| LocationX | System.Int16 | The X-coordinate of the upper-left corner of the image to display (pixels) |
| LocationY | System.Int16 | The Y-coordinate of the upper-left corner of the image to display (pixels) |
| Prompt | System.String | Prompt to display on the Cisco IP phone |
| Title | System.String | Title to display on the Cisco IP phone |
| URL | System.String | URL of the PNG image to download to the phone |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | Metreos.Types. CiscoIpPhone. CiscoIPPhoneImageFileType | Represents a minimal CiscoIpPhone.ImageFile message. This value can then be aggregated by a AddSoftKeyItem |

**Return Value**

Returns a properly formatted Cisco IP Phone file image display. The **`ResultData`** parameter should be assigned to a variable of type **`Metreos.Types.CiscoIpPhone.ImageFile`**.

**Remarks**

This action creates a properly formatted **`ImageFile`** object to be displayed on a Cisco IP Phone.

This action creates a file image object. The **`URL`** parameter must point to a PNG image within the bounds of what is displayable on the phone.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoIpPhone.CreateGraphicMenu Action

Creates a graphic menu display

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Data | System.String | Byte array of the data comprising the image |
| Depth | System.UInt16 | The number of bits used to determine the colors available for the image |
| Height | System.UInt16 | The height of the image (pixels) |
| LocationX | System.Int16 | The X-coordinate of the upper-left corner of the image to display (pixels) |
| LocationY | System.Int16 | The Y-coordinate of the upper-left corner of the image to display (pixels) |
| Prompt | System.String | Prompt to display on the Cisco IP phone |
| Title | System.String | Title to display on the Cisco IP phone |
| Width | System.UInt16 | The width of the image (pixels) |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | Metreos.Types.CiscoIpPhone.CiscoIPPhoneGraphicMenuType | Represents a minimal Cisco IP Phone GraphicMenu message. This value can then be aggregated by a AddMenuItem and a AddSoftKeyItem |

**Return Value**

Returns a properly formatted Cisco IP Phone graphic menu display. The `ResultData` parameter should be assigned to a variable of type `Metreos.Native.CiscoIpPhone.GraphicMenu`.

**Remarks**

This action creates the graphic menu object and fills in only the basic properties (including image data). Individual menu items can be added by calling **AddMenuItem** and specifying the resultant menu object of this operation as the target of the **AddMenuItem** action.

| |
|---|
| The `Data` parameter of the **`CreateGraphicMenu`** action must be a properly formatted CIP image. Refer to the 'Cisco IP Phone SDK' for more information on the CIP format. |
| **Requirements** |
| Metreos Framework version 2.0 |

## Metreos.Native.CiscoIpPhone.CreateGraphicFileMenu Action

Creates a graphic file menu display (models 7970 or greater only)

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Date Type | Description |
|---|---|---|
| LocationX | System.Int16 | The X-coordinate of the upper-left corner of the image to display (pixels) |
| LocationY | System.Int16 | The Y-coordinate of the upper-left corner of the image to display (pixels) |
| Prompt | System.String | Prompt to display on the Cisco IP phone |
| Title | System.String | Title to display on the Cisco IP phone |
| URL | System.String | URL of the PNG image to download to the phone |

**Result Data Fields**

| Parameter Name | Date Type | Description |
|---|---|---|
| ResultData | Metreos.Types.CiscoIpPhone. CiscoIPPhoneGraphicFileMenuType | Represents a minimal Cisco IP Phone graphic file menu message. This value can then be aggregated by a AddMenuItem and a AddSoftKeyItem |

**Return Value**

Returns a properly formatted Cisco IP Phone graphic file menu display. The **`ResultData`** parameter should be assigned to a variable of type **`Metreos.Types.CiscoIpPhone.GraphicFileMenu`**.

**Remarks**

This action creates a graphic menu object and fills in only the basic properties (including image data). Individual menu items can be added by calling **`AddMenuItem`** and specifying the resultant menu object of this operation as the target of the **`AddMenuItem`** action.

The **`URL`** parameter must point to a properly formatted PNG image.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoIpPhone.CreateMenu Action

Creates a menu display.

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Prompt | System.String | Prompt to display on the Cisco IP phone |
| Title | System.String | Title to display on the Cisco IP phone |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | Metreos.Types.CiscoIpPhone.CiscoIPPhoneMenuType | Represents a minimal Cisco IP Phone menu message. This value can then be aggregated by a **AddMenuItem** and a **AddSoftKeyItem** |

**Return Value**

Returns a properly formatted Cisco IP Phone menu display. The **`ResultData`** parameter should be assigned to a variable of type **`Metreos.Types.CiscoIpPhone.Menu`**.

**Remarks**

This action creates a menu object and fills in only the basic properties. Individual menu items can be added by calling **`AddMenuItem`** and specifying the resultant menu object of this operation as the target of the **`AddMenuItem`** action.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoIpPhone.CreateText Action

Creates a simple text display.

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Prompt | System.String | Prompt to display on the Cisco IP phone |
| Text | System.String | The text to display on the Cisco IP phone |
| Title | System.String | Title to display on the Cisco IP phone |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | Metreos.Types.CiscoIpPhone.CiscoIPPhoneTextType | Represents a minimal CiscoIpPhone.Text message. This value can then be aggregated by a AddSoftKeyItem |

**Return Value**

Returns a properly formatted Cisco IP Phone text screen. The **`ResultData`** parameter should be assigned to a variable of type **`Metreos.Types.CiscoIpPhone.Text`**.

**Remarks**

This action creates a simple text display. It utilizes the three main fields on the phone (title, prompt and main text area) to allow basic formatting of the message.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoIpPhone.SendExecute Action

Sends an execute command to the phone.

**Operation**

Synchronous

**Required Parameters**

None

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Message | System.String | A Cisco IP Phone execute object |
| Password | System.String | The password for the specified username |
| URL | System.String | The IP address of a Cisco IP phone. |
| Username | System.String | A valid username associated with the specified phone |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | System.Object | Represents a Cisco IP Phone result message. |

**Return Value**

Returns `success` if the phone was reachable and login was successful. Otherwise, `failure` is returned.

The two most common failure scenarios are if the phone is not reachable at the given URL or if the phone returned an error after receiving the request.

**Remarks**

This action sends a **`Metreos.Types.CiscoIpPhone.Execute`** object to a Cisco IP phone by way of an HTTP **PUT** request. The phone will then fetch the resource indicated by the URL in the Execute object.

The response received from the phone is parsed, placed in a **`Metreos.Types.CiscoIpPhone.Result`** object and passed back to the application.

**Requirements**

Metreos Framework version 2.0

### 2.7.2. Types

## Metreos.Types.CiscoIpPhone.DirectoryType

Represents a directory display.

**Input Types**

| Type Name | Description |
|---|---|
| CiscoIPPhoneDirectoryType | The result of a **CiscoIpPhone.CreateDirectory** action |
| CiscoIPPhoneDirectoryEntryType | The result of a **CiscoIpPhone.AddDirectoryEntry** action |
| CiscoIPPhoneSoftKeyType | The result of a **CiscoIpPhone.AddSoftKeyItem** action |

**Accessible Public Methods**

None.

**Accessible Public Properties**

None.

**Remarks**

Once initialized and populated, send this type as the body of the HTTP response to the phone's request. Be sure to set the **`Content-Type`** header of the response to **text/xml**. Also, best practice dictates that the HTTP **`Expires`** header should be set to **`-1`**.

**Requirements**

Metreos Framework version 2.0

## Metreos.Types.CiscoIpPhone.GraphicFileMenu Type

Represents a graphic file menu display.

**Input Types**

| Type Name | Description |
|---|---|
| CiscoIPPhoneGraphicFileMenuType | The result of a **CiscoIpPhone.CreateGraphicFileMenu** action |
| CiscoIPPhoneMenuItemType | The result of a **CiscoIpPhone.AddMenuItem** action |
| CiscoIPPhoneSoftKeyType | The result of a **CiscoIpPhone.AddSoftKeyItem** action |

**Accessible Public Methods**

None.

**Accessible Public Properties**

None.

**Remarks**

Once initialized and populated, send this type as the body of the HTTP response to the phone's request. Be sure to set the `Content-Type` header of the response to **text/xml**. Also, best practice dictates that the HTTP **`Expires`** header should be set to **`-1`**.

**Requirements**

Metreos Framework version 2.0

# Metreos.Types.CiscoIpPhone.GraphicMenu Type

Represents a graphic menu display.

**Input Types**

| Type Name | Description |
|---|---|
| CiscoIPPhoneGraphicMenuType | The result of a **CiscoIpPhone.CreateGraphicMenu** action |
| CiscoIPPhoneMenuItemType | The result of a **CiscoIpPhone.AddMenuItem** action |
| CiscoIPPhoneSoftKeyType | The result of a **CiscoIpPhone.AddSoftKeyItem** action |

**Accessible Public Methods**

None.

**Accessible Public Properties**

None.

**Remarks**

Once initialized and populated, send this type as the body of the HTTP response to the phone's request. Be sure to set the `Content-Type` header of the response to **text/xml**. Also, best practice dictates that the HTTP **`Expires`** header should be set to **`-1`**.

**Requirements**

Metreos Framework version 2.0

# Metreos.Types.CiscoIpPhone.IconMenu Type

Represents an icon menu display.

**Input Types**

| Type Name | Description |
|---|---|
| CiscoIPPhoneIconMenuType | The result of a **CiscoIpPhone.CreateIconMenu** action |
| CiscoIPPhoneMenuItemType | The result of a **CiscoIpPhone.AddMenuItem** action |
| CiscoIPPhoneSoftKeyType | The result of a **CiscoIpPhone.AddSoftKeyItem** action |
| CiscoIPPhoneIconItemType | The result of a **CiscoIpPhone.AddIconItem** action |

**Accessible Public Methods**

None.

**Accessible Public Properties**

None.

**Remarks**

Once initialized and populated, send this type as the body of the HTTP response to the phone's request. Be sure to set the **`Content-Type`** header of the response to **text/xml**. Also, best practice dictates that the HTTP **`Expires`** header should be set to **`-1`**.

**Requirements**

Metreos Framework version 2.0

# Metreos.Types.CiscoIpPhone.Image Type

Represents an image display.

**Input Types**

| Type Name | Description |
|---|---|
| CiscoIPPhoneImageType | The result of a **CiscoIpPhone.CreateImage** action |
| CiscoIPPhoneSoftKeyType | The result of a **CiscoIpPhone.AddSoftKeyItem** action |

**Accessible Public Methods**

None.

**Accessible Public Properties**

None.

**Remarks**

Once initialized and populated, send this type as the body of the HTTP response to the phone's request. Be sure to set the **`Content-Type`** header of the response to **text/xml**. Also, best practice dictates that the HTTP **`Expires`** header should be set to **`-1`**.

**Requirements**

Metreos Framework version 2.0

# Metreos.Types.CiscoIpPhone.Directory Type

Represents an image file display.

**Input Types**

| Type Name | Description |
|---|---|
| CiscoIPPhoneImageFileType | The result of a **CiscoIpPhone.CreateImage** action |
| CiscoIPPhoneSoftKeyType | The result of a **CiscoIpPhone.AddSoftKeyItem** action |

**Accessible Public Methods**

None.

**Accessible Public Properties**

None.

**Remarks**

Once initialized and populated, send this type as the body of the HTTP response to the phone's request. Be sure to set the **`Content-Type`** header of the response to **text/xml**. Also, best practice dictates that the HTTP **`Expires`** header should be set to **`-1`**.

**Requirements**

Metreos Framework version 2.0

# Metreos.Types.CiscoIpPhone.Input Type

Represents an input prompt.

**Input Types**

| Type Name | Description |
|---|---|
| CiscoIPPhoneInputType | The result of a **CiscoIpPhone.CreateInput** action |
| CiscoIPPhoneInputItemType | The result of a **CiscoIpPhone.AddInputItem** action |
| CiscoIPPhoneSoftKeyType | The result of a **CiscoIpPhone.AddSoftKeyItem** action |

**Accessible Public Methods**

None.

**Accessible Public Properties**

None.

**Remarks**

Once initialized and populated, send this type as the body of the HTTP response to the phone's request. Be sure to set the **`Content-Type`** header of the response to **text/xml**. Also, best practice dictates that the HTTP **`Expires`** header should be set to **`-1`**.

**Requirements**

Metreos Framework version 2.0

# Metreos.Types.CiscoIpPhone.Menu Type

Represents a simple menu display.

**Input Types**

| Type Name | Description |
|---|---|
| CiscoIPPhoneMenuType | The result of a **CiscoIpPhone.CreateMenu** action |
| CiscoIPPhoneMenuItemType | The result of a **CiscoIpPhone.AddMenuItem** action |
| CiscoIPPhoneSoftKeyType | The result of a **CiscoIpPhone.AddSoftKeyItem** action |

**Accessible Public Methods**

None.

**Accessible Public Properties**

None.

**Remarks**

Once initialized and populated, send this type as the body of the HTTP response to the phone's request. Be sure to set the **`Content-Type`** header of the response to **text/xml**. Also, best practice dictates that the HTTP **`Expires`** header should be set to **`-1`**.

**Requirements**

Metreos Framework version 2.0

# Metreos.Types.CiscoIpPhone.Text Type

Represents a text display.

**Input Types**

| Type Name | Description |
|---|---|
| CiscoIPPhoneTextType | The result of a **CiscoIpPhone.CreateText** action |
| CiscoIPPhoneSoftKeyType | The result of a **CiscoIpPhone.AddSoftKeyItem** action |

**Accessible Public Methods**

None.

**Accessible Public Properties**

None.

**Remarks**

Once initialized and populated, send this type as the body of the HTTP response to the phone's request. Be sure to set the **`Content-Type`** header of the response to **text/xml**. Also best practice dictates that the HTTP `Expires` header should be set to `-1`.

**Requirements**

Metreos Framework version 2.0

# Metreos.Types.CiscoIpPhone.Response Type

Represents the response to a **CiscoIpPhone.SendExecute** action.

**Input Types**

| Type Name | Description |
|---|---|
| CiscoIPPhoneResponseType | One possible result of a **CiscoIpPhone.SendExecute** action |
| CiscoIPPhoneErrorType | The other possible result of **a CiscoIpPhone.SendExecute** action |

**Accessible Public Methods**

None.

**Accessible Public Properties**

| Type Name | Description |
|---|---|
| System.Boolean IsError | Returns a boolean indicated whether the phone responded with an error. |

**Remarks**

In this release, this type only identifies if an error occurred, but it does not return the actual error text. It is possible however to forward the response as XML to a client for display by serializing this type in the body of an HTTP **POST** request.

**Requirements**

Metreos Framework version 2.0

# Metreos.Types.CiscoIpPhone.Execute Type

Represents a Cisco IP Phone Execute command.

**Input Types**

| Type Name | Description |
|---|---|
| CiscoIPPhoneExecuteType | The result of a **CiscoIpPhone.CreateExecute** action |

**Accessible Public Methods**

None.

**Accessible Public Properties**

None.

**Remarks**

This type differs from the other Cisco IP phone types in that it is sent in the body of an HTTP request instead of a response. Use the **`SendExecute`** action to send the request to the phone.

**Requirements**

Metreos Framework version 2.0

## 2.8. Timer Facility

### 2.8.1. Actions

#### Metreos.Providers.TimerFacility.AddTriggerTimer Action

Add a new low-resolution timer which will trigger a new script instance.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| timerDateTime | System.DateTime | Initial time to fire. Must be in a .NET Framework **System.DateTime** parseable format |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| timerRecurrenceInterval | System.TimeSpan | Period of the timer. Must be in a .NET Framework **System.TimeSpan** parseable format |
| timerUserData | System.String | An opaque token used to allow distinguishable timer events to be raised |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| timerId | System.String | Unique timer identifier |

**Return Value**

Returns **`success`** if the timer was added; otherwise, **`failure`** is returned.

**Remarks**

Adds a new timer to the timer table of the Timer Facility Provider. This action will fire **`Metreos.Providers.TimerFacility.TimerFired`** triggering a new script.

Applications may specify a recurrence interval for the timer. This is a period of time over which the timer will repeat until removed by the application. Further, **`timerUserData`** will be delivered to the application in the **`TimerFire`** event.

**Requirements**

Metreos Framework version 2.0

Metreos Timer Facility Provider version 2.0

## Metreos.Providers.TimerFacility.AddNonTriggerTimer Action

Add a new low-resolution timer that will fire an event in the script instance which invoked this action.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| timerDateTime | System.DateTime | Initial time to fire. Must be in a .NET Framework **System.DateTime** parseable format |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| timerRecurrenceInterval | System.TimeSpan | Period of the timer. Must be in a .NET Framework **System.TimeSpan** parseable format |
| timerUserData | System.String | An opaque token used to allow distinguishable timer events to be raised |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| timerId | System.String | Unique timer identifier |

**Return Value**

Returns **`success`** if the timer was added; otherwise, **`failure`** is returned.

**Remarks**

Adds a new timer to the timer table of the Timer Facility Provider. The timer, when fired, will raise a **`Metreos.Providers.TimerFacility.TimerFired`** event in the script instance which called **`Metreos.Providers.TimerFacility.AddNonTriggerTimer`**.

Applications may specify a recurrence interval for the timer. This is a period of time over which the timer will repeat until removed by the application. Further, **`timerUserData`** will be delivered to the application in the **`TimerFire`** event.

**Requirements**

Metreos Framework version 2.0

Metreos Timer Facility Provider version 2.0

## Metreos.Providers.TimerFacility.RemoveTimer Action

Remove a timer that has been previously added by an application.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| timerId | System.String | Unique timer identifier |

**Optional Parameters**

None

**Result Data Fields**

None

**Return Value**

Returns `success` if the timer was successfully removed; otherwise, `failure` is returned.

**Remarks**

Timers that have a recurrence interval specified will continue to fire until removed by the application that added them. Further, it is a good practice to always remove a timer even if no recurrence interval is specified. The **`timerId`** parameter indicates which timer should be removed from the Timer Facility Provider's timer table.

**Requirements**

Metreos Framework version 2.0

Metreos Timer Facility Provider version 2.0

### 2.8.2. Events

## Metreos.Providers.TimerFacility.TimerFire Event

Event fired when a previously added timer is triggered.

**Type**

Non-Triggering

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| timerId | System.String | Unique timer identifier |
| timerUserData | System.String | An opaque token used to allow distinguishable timer events to be raised |

**Remarks**

The **`TimerFire`** event is fired by the Timer Facility Provider when a previously added timer is triggered. The event will contain both the ID of the timer that has fired along with the user data specified in the **`AddTimer`** action.

**Requirements**

Metreos Framework version 2.0

Metreos Timer Facility Provider version 2.0

## 2.9. Cisco DeviceListX

The Cisco DeviceListX provider included with the MCE allows the device list data to be cached locally on the Metreos Application Runtime Environment. This lets running applications to query the device list data without concern for impacting the performance of their Cisco CallManager.

### 2.9.1. Actions

**Metreos.Providers.CiscoDeviceListX.Refresh Action**

The Refresh action causes the DeviceListX provider to initiate a manual refresh of the device list data cache.

**Operation**

Asynchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application |

**Optional Parameters**

None.

**Result Data Fields**

None.

**Return Value**

Returns a provisional response of **`success`** if the DeviceListX provider was able to initiate a refresh of the device list data cache, otherwise **`failure`** is returned.

A final response will be returned to the application in the form of an event when the device list data cache refresh has completed. A **`Refresh_Complete`** event indicates a successful final response. Likewise, a **`Refresh_Failed`** event indicates that an error occurred and the refresh was unable to complete.

**Remarks**

When the DeviceListX provider initiates a device list data cache refresh it must contact all of the configured CallManager subscribers and request a new device list data dump. This could potentially take a very long time and applications must be aware of this when using the **`Refresh`** action.

To reduce the impact to Cisco CallManager the DeviceListX provider will not process back-to-back refresh requests. In this situation, requests that were not processed will be indicated

| |
|---|
| as having failed. Furthermore, if an automatic refresh has been recently completed the manual refresh request may result in a failure. |
| **Requirements** |
| Metreos Framework version 2.0<br>Metreos Cisco DeviceListX Provider version 2.0 |

## Metreos.Providers.CiscoDeviceListX.Refresh Action

The Refresh action causes the DeviceListX provider to initiate a manual refresh of the device list data cache, but does not

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application |

**Optional Parameters**

None.

**Result Data Fields**

None.

**Return Value**

Returns a provisional response of `success` if the DeviceListX provider was able to initiate a refresh of the device list data cache, otherwise **`failure`** is returned.

**Remarks**

Because the a DeviceListX Refresh can take a very long time, this synchronous version of the `Refresh` exists to offer a way to refresh without caring, as an application developer, when the **`Refresh`** fully completes.

**Requirements**

Metreos Framework version 2.0

Metreos Cisco DeviceListX Provider version 2.0

## Metreos.Native.CiscoDeviceList.Query Action

Queries the Cisco DeviceListX cache for device information.

**Operation**

Synchronous

**Required Parameters**

None.

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Description | System.String | Description of the device |
| IP | System.String | IP address of the device |
| Name | System.String | Name of the device |
| Pool | System.String | Device pool in which this device is contained |
| SearchSpace | System.String | Search space of the device |
| Status | System.String | Status of the device |
| Type | System.String | Type code of the device |

**Result Data Fields**

| Parameter Name | Description |
|---|---|
| ResultData | System.Data.DataTable |

**Return Value**

Returns `success` if the database query completed without errors; otherwise, `failure`. Note that a successful response does not necessarily mean that any devices were found matching the specified criteria.

**Remarks**

This action will use all the optional parameters supplied to it in a logical **AND** manner to locate a matching device definition. Once the action has completed successfully and the **`ResultData`** result data field has been assigned to a variable, the rows can be dealt with as individual **`DataRow`** objects, and one can access the individual columns with the string values: `Type`, `Name`, **`Description`**, **`SearchSpace`**, **`Pool`**, **`IP`**, **`Status`**.

**Requirements**

Metreos Framework version 2.0

Metreos Cisco DeviceListX Provider version 2.0

### 2.9.2. Asynchronous Callback Events

## Metreos.Providers.CiscoDeviceListX.Refresh_Complete Event

Event fired to the application indicating successful completion of a previously initiated `Refresh` action.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application |

**Remarks**

None.

**Requirements**

Metreos Framework version 2.0

Metreos Cisco DeviceListX Provider version 2.0

## Metreos.Providers.CiscoDeviceListX.Refresh_Failed Event

Event fired to the application indicating failure of a previously initiated `Refresh` action.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application |

**Remarks**

None.

**Requirements**

Metreos Framework version 2.0

Cisco DeviceListX Provider version 2.0

## 2.10. Cisco Extension Mobility

The Cisco Extension Mobility API exposed by Cisco CallManager enables a third-party to control the Extension Mobility functionality in CallManager. You can dynamically log users in to and out of devices, and query CallManager for device-user and user-device associations.

### Metreos.Native.CiscoExtensionMobility.Login Action

Logs in a user into a given device with a default or specified profile.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| AppId | System.String | The username of a user with authentication proxy rights enabled |
| AppCertificate | System.String | The password of a user with authentication proxy rights enabled |
| UserId | System.String | The user to log in |
| DeviceName | System.String | The name of the device (SEP + MAC) |
| DeviceProfile | System.String | The name of the profile to use |
| Url | System.String | Login URL for the Extension Mobility Service |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Timeout | System.Int32 | Amount of time in minutes to lease this device for |
| NoTimeout | System.Boolean | Set to **false** if Timeout is to be used |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ErrorCode | System.Int32 | The Cisco-specific error code if the Extension Mobility service failed to login the user. If **0** is returned on failure, then the action was never able to communicate with CallManager |
| ErrorMessage | System.String | A Cisco-specific error message returned by the Extension Mobility service. |

**Return Value**

Returns **`success`** if the user could be logged in using Extension Mobility. Otherwise, **`failure`** is returned.

**Remarks**

The **`AppId`** and **`AppCertificate`** are any username and password inside CallManager which have proxy rights enabled. This `Username` can be the same as specified for `UserId`. If no **`DeviceProfile`** is specified, the user's default device profile is used.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoExtensionMobility.Logout Action

Logouts out any user logged into a given device.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| AppId | System.String | The username of a user with authentication proxy rights enabled |
| AppCertificate | System.String | The password of a user with authentication proxy rights enabled |
| DeviceName | System.String | The name of the device (SEP + MAC) |
| Url | System.String | Logout URL for the Extension Mobility Service |

**Optional Parameters**

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ErrorCode | System.Int32 | The Cisco-specific error code if the Extension Mobility service failed to login the user. If **0** is returned on failure, then the action was never able to communicate with CallManager |
| ErrorMessage | System.String | A Cisco-specific error message returned by the Extension Mobility service. If it is null, then the action never communicated with CallManager |

**Return Value**

Returns **`success`** if the device could be logged out using Extension Mobility. Otherwise, **`failure`** is returned.

**Remarks**

The **`AppId`** and **`AppCertificate`** are any username and password inside CallManager which have proxy rights enabled.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoExtensionMobility.QueryDevices Action

Queries a number of devices, returning which users, if any, own them.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| AppId | System.String | The username of a user with authentication proxy rights enabled |
| AppCertificate | System.String | The password of a user with authentication proxy rights enabled |
| DeviceNames | System.String[] | The name(s) of the device(s). |
| Url | System.String | Query URL for the Extension Mobility Service |

**Optional Parameters**

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| QueryDevicesResult | Metreos.Types.<br>CiscoExtensionMobility.<br>UserResponse | A complex variable type containing the data returned by this action. Is populated only on success |
| ErrorMessage | System.String | A Cisco-specific error message returned by the Extension Mobility service. If it is null, then the action never communicated with CallManager |

**Return Value**

Returns **`success`** if the devices could be queried using Extension Mobility. Otherwise, **`failure`** is returned.

**Remarks**

The **`AppId`** and **`AppCertificate`** are any username and password inside CallManager which have proxy rights enabled. The **`QueryDevicesResult`** variable contains complex information regarding the information returned by the Extension Mobility; because it is not easy to parse using **`CustomCode`**, the **`GetDeviceStatus`** action was designed to help retrieving data from that variable.

**Requirements**

Metreos Framework version 2.0

## Metreos.Native.CiscoExtensionMobility.GetDeviceStatus Action

Finds the status of a device, returning the user logged into that device if applicable.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| DeviceName | System.String | The name of the device (SEP + MAC) |
| QueryDeviceResults | Metreos.Types.<br>CiscoExtensionMobility.<br>QueryDeviceResults | The results from a QueryDevices action |

**Optional Parameters**

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| Username | System.String | If the device is **LoggedIn** status, the username associated with this device |

**Return Value**

Return **`NoDevice`** if the device could not be found in the results. **`NoUser`** is returned if the device could be found, but has no user associated. **`LoggedIn`** is returned if the device could be found and if it has a user associated with it as well. A **`failure`** is returned if the **`QueryDeviceResults`** parameter was returned from a **`QueryDevices`** action which failed.

**Remarks**

`Username` is only valid if the action returns **`LoggedIn`**.

**Requirements**

Metreos Framework version 2.0

### Metreos.Native.CiscoExtensionMobility.ValidatePin Action

Validates the pin of a user in an LDAP server that conforms to the Cisco CallManager schema.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Username | System.String | The username of the user to validate |
| Pin | System.String | The pin of the user to validate |
| LdapServerHost | System.String | The address of the LDAP server |
| LdapServerPost | System.Uint32 | The port of the LDAP server |
| LdapUsername | System.String | The LDAP administrative username |
| LdapPassword | System.String | The LDAP administrative password |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CustomUserAttr | System.String | If the username attribute is not in the 'cn' attribute, use this field to override that value to search on |
| LdapBaseDn | System.String | Base LDAP DN. The default is (ou=Users, o=cisco.com) |

**Result Data Fields**

**Return Value**

A **`success`** is returned if the user could be validated. An **`InvalidUsernamePin`** indicates that either the `Username` or `Pin` is invalid. A `failure` indicates that the LDAP query failed.

**Remarks**

In a default installation, it should not be necessary to install **`CustomUserAttr`** or **`LdapBaseDn`**.

**Requirements**

Metreos Framework version 2.0

## 2.11. Cisco AXL SOAP

Cisco AXL SOAP will be documented in 2.1.

## 2.12. Dial Plan

### 2.12.1. Actions

#### Metreos.Native.DialPlan.FormatAddress Action

This action applies the configured dial plan to the specified dialed number (DN).

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| DialedNumber | System.String | The number to format. Can be already fully qualified with the communications system's IP address |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| CM_Address | System.String | IP address of outbound gateway |
| DialingRules | System.Collections.Hashtable | A hash table containing patterns to apply to the fully qualified number, with replacement strings to apply to a matching substring, if one occurs. Keys are the patterns, values are the replacements |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| ResultData | System.String | The formatted address with all the dialing rules applied |

**Return Value**

Returns **`success`** if enough parameters where specified to successfully format the DN. Otherwise, **`failure`** is returned.

**Remarks**

This action will format a given DN into the *dialedNumber@serverAddress* notation. If the DN is already in that notation, the **CM_Address** parameter may be omitted. Otherwise, the **`CM_Address`** parameter must be present so that it can be suffixed to the *dialedNumber@*' portion of the address.

The **`DialingRules`** parameter is a hash table of regular expressions which represent the dialing rule to be applied. The hash table key is the pattern to match and the value is the

| |
|---|
| regular expression transformation to apply. |
| **Requirements** |
| Metreos Framework version 2.0 |

## 2.13. Mail

Actions in this namespace support sending SMTP email. Supported email functionality includes sending as HTML, one or more attachments, and SMTP authentication.

### 2.13.1. Actions

#### Metreos.Native.Mail.Send Action

Send an SMTP formatted email.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| To | System.String | A recipient for the email |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| AttachmentPaths | System.Collections.Specialized.StringCollection | A StringCollection of files to attach to the email |
| AuthenticationMode | System.String | Authentication mode mandated by mail server (default: Base64) |
| Body | System.String | The body of the email |
| From | System.String | Identity to use as sender |
| MailServer | System.String | Mail server to use if different than the domain of the recipient |
| Password | System.String | Mail server credentials |
| SendAsHtml | System.Boolean | Boolean indicating whether to send as HTML (default: false) |
| Subject | System.String | The subject of the email |
| Username | System.String | Mail server credentials |

**Result Data Fields**

None.

**Return Value**

Returns `success` if the mail was sent successfully, otherwise `failure`.

**Remarks**

This action sends an email to the specified recipient using the specified server. All basic SMTP functions are supported including authentication and attachments. The MIME type of the attachments is auto-detected.

The action must include **`Username`** and **`MailServer`** or `From` or both.

The action must also include **`Subject`** or **`Body`** or **`AttachmentPaths`** or any combination of these. Multiple recipients may be specified by separating their e-mail addresses with a comma (`,`).

**Requirements**

Metreos Framework version 2.0

## 2.14. Call Control (Deprecated)

### 2.14.1. Actions

#### Metreos.Providers.H323.AnswerCall Action

Answer an incoming 1st party phone call.

**Operation**

Asynchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| answer | System.Boolean | A boolean value indicating whether to answer the call or not. |
| callId | System.String | A unique token used to identify this particular call leg in future actions and events. |
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application. Leave this at the default 'none' unless you need to change this. |

**Optional Parameters**

None.

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Return Value**

Returns **`success`** if the call is being answered; otherwise, **`failure`** is returned.

**Remarks**

This action answers an incoming call based on the value of the boolean parameter `answer`. If `answer` is set to `true` the call will be answered normally. If answer is set to **`false`** the incoming call will be rejected. Once all negotiations have been completed with the caller an **`AnswerCall_Complete`** event will be fired to the application. If an error occurs and the call can not be answered an **`AnswerCall_Failed`** event will be fired to the application. If the application requires full-duplex media, invoke `SetMedia`.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

## Metreos.Providers.H323.Hangup Action

Terminate an existing first party phone call.

**Operation**

Asynchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application. Leave this at the default 'none' unless you need to change this. |

**Optional Parameters**

None.

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Return Value**

Returns **`success`** if the call is being terminated; otherwise, **`failure`** is returned.

**Remarks**

This action terminates the existing call with the specified call ID. This action should only be used with calls which have been successfully established and should not be used to reject incoming calls. To reject an incoming call, use **`AnswerCall`** with the answer parameter set to false.

Once the Application Runtime Environment has finalized the hang up a **`Hangup_Complete`** event will be fired to the application. If an error occurs and the call could not be terminated a **`Hangup_Failed`** event will be fired to the application.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

## Metreos.Providers.H323.MakeCall Action

Initiate a first party phone call from the Application Runtime Environment to a remote party.

**Operation**

Asynchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| mediaIP | System.String | The IP address of the media server that will be terminating media for this call |
| mediaPort | System.UInt16 | The port number on the media server where the media for this call should be sent |
| to | System.String | The fully-qualified address of the remote party to be called |
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application. Leave this at the default 'none' unless you need to change this. |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| from | System.String | A friendly name for the originator of this call (generally, this should be set to the application's name) |
| connectionId | System.Int32 | Media server connection ID associated with this call. |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Return Value**

Returns **`success`** if the call was initiated; otherwise, **`failure`** is returned.

**Remarks**

This action initiates a call from the Application Runtime Environment to the location specified in the `to` parameter. It is assumed that appropriate media resources have been reserved on the media server prior to this action being performed. The media details should be passed in as the `mediaIp` and `mediaPort` parameters.

One the Application Runtime Environment has fully established the call a

**MakeCall_Complete** event will be fired to the application. If the call can not be established then a **MakeCall_Failed** event will be fired to the application.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

## Metreos.Providers.H323.SetMedia Action

Set the media information to be used when answering an incoming 1st party phone call.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |
| mediaIP | System.String | The IP address of the media server that will be terminating media for this call |
| mediaPort | System.UInt16 | The port number on the media server where the media for this call should be sent |

**Optional Parameters**

None.

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Return Value**

Returns `success` if the information was set; otherwise, `failure` is returned.

**Remarks**

Use this action to inform the call control stack what IP address and port for RTP media to use when establishing an incoming call. This action should be called before each **AnswerCall**. Usually, the media values will be the result of a media server half-connect action (See the `CreateConnection` table in the *Media Control* section). If the call is going to be rejected, set the `mediaIp` and `port` parameters to 0.

This call is divorced from **AnswerCall**. It is imperative that this action be called as quickly as possible after the **IncomingCall** event has been received. **AnswerCall** should be called immediately because the provider will wait on it for a limited time. After that, there can be a longer delay before **AnswerCall** is executed.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

### 2.14.2. Events

## Metreos.Providers.H323.GotDigits Event

Event fired indicating that DTMF digits have been observed on the call signaling path.

**Type**

Non-Triggering

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |
| digits | System.String | A string of DTMF digits |

**Remarks**

This event will fire for each digit received over the call control signaling channel. For these digits to affect media processing termination conditions those digits must be inserted into the media processing stream.

See **`Metreos.Providers.MediaServer.SendDigits`** in the *Media Control* section for more details.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

# Metreos.Providers.H323.Hangup Event

Event fired indicating that an existing first party phone call has ended without the application's request.

**Type**

Non-Triggering

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |
| from | System.String | The directory number of the remote calling party |
| reason | System.String | The reason the call was terminated |
| to | System.String | The dialed number of the local party being called |

**Remarks**

When this event is received, the call has been terminated. No further action is necessary. The **`callId`** parameter indicates which call leg has been terminated. The **`reason`**, **`to`**, and **`from`** parameters are intended only for convenience when displaying output to a human user.

Generally this event will fire in response to the remote party physically placing their terminal on hook; however, it is possible for this event to fire if the underlying call **control** stack terminates the call for network or other call signaling reasons. In this situation, the **`reason`** parameter will indicate why the call was terminated. It is returned in certain **`Metreos.Providers.H323`** events and has one of five values, each of which correspond to the cause of the event. The events that return a reason parameter are **`MakeCall_Failure`** and **`Hangup`**.

Reason codes and list of possible conditions which could have caused each reason code:

**`NormalCallClearing`**

- Ended By Local User
- Ended By Remote User
- Ended By Caller Abort

**`NoAnswer`**

- Ended By No Answer
- Ended By Answer Denied
- Ended By Refusal

**`RemoteBusy`**

- Ended By Remote Busy

**`Unreachable`**

- Ended By Unreachable

- Ended By Local Congestion
- Ended By Host Offline
- Ended By No Bandwidth
- Ended By No User
- Ended By Remote Congestion

**`OtherOrUnknown`**

- All other cases

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

## Metreos.Providers.H323.IncomingCall Event

Event fired indicating a remote party is attempting to establish a first party phone call with the Application Runtime Environment.

**Type**

Triggering

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |
| from | System.String | The directory number of the remote calling party |
| to | System.String | The dialed number of the local party being called |

**Remarks**

This event signals the beginning of a call. It provides the remote party's address as well as the local identity that party is trying to reach. Both **`SetMedia`** and **`AnswerCall`** actions should be invoked in response to this event.

If the call is intended to be answered, the normal case is to perform a **half-connect** operation on the media server to reserve the necessary media resources. To do so, execute **`Metreos.Provider.MediaServer.CreateConnection`** with the remote party's media IP, port, and connectionId set to **0**. Then, use **`SetMedia`** to inform the call control provider of these resources. Next, **`AnswerCall`** should be executed with the `answer` parameter set to **`true`**.

If the call is to be rejected, **`SetMedia`** should be executed with all media information set to **0**. Then, **`Answer`** should be executed with the `answer` parameter set to **`false`**.

The **`callId`** parameter contains the unique token that should be used when performing any actions on this call leg. For instance, if the application is going to terminate a call, then a **`Hangup`** action would be invoked with the **`callId`** from this event.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

## Metreos.Providers.H323.SignalingChange Event

Event fired indicating that the remote party has re-negotiated their media parameters.

**Type**

Non-Triggering

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |
| mediaIP | System.String | The IP address of the remote party that will be terminating media for this call |
| mediaPort | System.Int32 | The port number on the remote party where the media for this call should be sent |

**Remarks**

This event will occur only after a call has been successfully established. It is typically fired in response to a call signaling message indicating that the remote party has

- Placed the call on hold
- Taken the call off hold
- Transferred the call to a different endpoint.

In response to this event the application would typically send the new media information to the media server so that the full-duplex media channel may be preserved. Execute a `Metreos.Providers.MediaServer.CreateConnection` action and specify the connection ID which was returned from the initial invocation of the action along with the new media parameters. Refer to the *Media Control* section for more details.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

### 2.14.3. Asynchronous Callback Events

## Metreos.Providers.H323.AnswerCall_Complete Event

Event fired indicating that the **`AnswerCall`** action completed successfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |
| from | System.String | The directory number of the remote calling party |
| to | System.String | The dialed number of the local party being called |

**Remarks**

This event indicates that an inbound, first party call has been fully connected to the Application Runtime Environment. Applications may perform further actions on this call leg by invoking call control actions with the **`callId`** parameter set to the value received in this event.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

## Metreos.Providers.H323.AnswerCall_Failed Event

Event fired indicating that the **`AnswerCall`** action failed to complete.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Remarks**

The call control stack was unable to finish answering the call. No further action is required by the application. The application should no longer attempt call control actions on this call leg as the value of the **`callId`** parameter is no longer an active call.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

## Metreos.Providers.H323.Hangup_Complete Event

Event fired indicating that the `Hangup` action completed successfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Remarks**

The hang up operation has completed successfully. Any state related to the call can now be safely cleaned up. The application should no longer attempt call control actions on this call leg as the value of the **`callId`** parameter is no longer an active call.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

## Metreos.Providers.H323.Hangup_Failed Event

Event fired indicating that the **`Hangup`** action completed unsuccessfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |

**Remarks**

The call control stack was unable to hang up the specified call leg. Upon receiving this event the application should no longer reference the **`callId`** that was used in the original hang up operation and should consider the call to be terminated.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

## Metreos.Providers.H323.MakeCall_Complete Event

Event fired indicating that the **`MakeCall`** action completed successfully.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |
| mediaIP | System.String | The IP address of the remote party that will be terminating media for this call |
| mediaPort | System.String | The port number on the remote party where the media for this call should be sent |

**Remarks**

This event indicates successful connection with the remote party for an Application Runtime Environment initiated, first party call. Normally, the application would initiate a 'full connect' to the media server at this time by executing **`Metreos.Providers.MediaServer.CreateConnection`** with the remote party's media information filled in from the parameters of this event. Once that is completed, media will be flow bidirectionally between both devices.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

## Metreos.Providers.H323.MakeCall_Failed Event

Event fired indicating that the `MakeCall` action failed to complete.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| callId | System.String | A unique token used to identify this particular call leg in future actions and events |
| from | System.String | The directory number of the remote calling party |
| reason | System.String | The reason the call was terminated. |
| to | System.String | The dialed number of the local party being called |

**Remarks**

The **`MakeCall`** action has failed due to a rejection by the remote party or ring out conditions. No further action is required. The application should no longer attempt call control actions on this call leg as the value of the **`callId`** parameter is no longer an active call.

Refer to the description of the **`Hangup`** event for an explanation of and the possible values for the **`reason`** parameter.

**Requirements**

Metreos Framework version 2.0

Metreos H.323 Provider version 2.0

## 2.15. Media Control (Deprecated)

The Media Engine responds with a numeric result code after completing a request. The following table enumerates these codes:

| Result Code | Reason | Description |
|---|---|---|
| 0 | Success | The action completed successfully |
| 1 | Provisional OK | The action is executing asynchronously |
| 4 | All sessions are in use | The maximum configured number of connections has been reached |
| 5 | Server not accepting commands | Media server is in shutdown state |
| 6 | Internal error | Internal logic error |
| 7 | Device error | Media firmware error |
| 8 | Media resource not available | All conference or voice resources are busy |
| 9 | Internal error | Internal call state transition error |
| 10 | Internal error | Internal event table i/o error |
| 11 | Internal error | Internal asynchronous event state error |
| 12 | Session timeout | No activity on session within the configured timeout interval |
| 13 | Action timeout | An action did not complete within the configured timeout interval |
| 14 | Session busy | The connection is busy with another request |
| 15 | Already connected | A connection with the supplied ID already exists |
| 16 | No connection exists | A Media server operation was requested for a connection ID which does not exist |
| 20 | Unrecognized event | Internal error: unexpected media termination event |
| 21 | Nonexistent command | An invalid command ID was specified |
| 22 | Connection ID invalid format | The specified connection ID was not well-formed, e.g. negative |
| 23 | Connection ID not registered | The specified ID does not match any current connection |
| 24 | Session is not in conference | A conference operation was requested on a connection which is not in conference |
| 25 | Reserved for future use | |
| 26 | Too few parameters supplied | Not all expected parameters were supplied |
| 27 | Value error | The value of an action parameter was outside the parameter's valid range of values |
| 30 | File open error | A specified voice file could not be opened |
| 31 | File read or write error | A specified voice file could not be read |
| 35 | Malformed request | A protocol block used to transport a request to the Media server was malformed |
| 36 | Invalid termination condition | Unrecognized termination condition supplied |

### 2.15.1. Actions

## Metreos.Providers.MediaServer.CreateConnection Action

Create a voice-capable connection to the media server. This action can be used to reserve a port on a media server for later use.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| connectionId | System.Int32 | Connection ID associated with this connection |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| remoteIp | System.String | IP address of the remote media endpoint |
| remotePort | System.UInt16 | Port on which the remote media endpoint is listening for media |
| connectionAttribute | System.String | Any additional conference attributes can be specified in this field |
| callId | System.String | Call ID associated with this connection |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| connectionId | System.String | Unique Media server ID for a connection |
| ipAddress | System.String | Local media IP address |
| port | System.String | Local media port |
| resultCode | System.String | Result of a Media server operation |

**Return Value**

Returns **`success`** if the connection was successfully created. Otherwise **`failure`** is returned. If the media server fails to create the connection, the **`resultCode`** parameter will hold an integer value indicating the reason why the connection could not be created.

**Remarks**

Applications that use media capabilities exposed by the MCE must first create a connection to a media server. Media server connections are specific to a distinct user of an application. If, for example a conferencing application conferences four individual phone calls, four distinct media server connections must be created first, and then put into the same conference.

Some applications reserve a media port before knowing the remote party's media details.

Also known as a half-connection, reserving media allows the application to ensure a media port is available. Reserving media also retrieves the media attributes for passing on to the remote party of the application. To execute a half-connection issue a **`CreateConnection`** action with **`0`** as the **`connectionId`**, **`remoteIp`** and **`remotePort`** parameters. The media server then reserves a port and returns the **`ipAddress`** and **`port`** number for the new connection.

Half-connections allow you to pass on media server information to the remote party, effectively setting up a half-duplex connection. To transition to a full-duplex connection issue a second **`CreateConnection`** action, this time passing in the **`connectionId`** returned in the original **`CreateConnection`**. The remote party must also be passed into **`remoteIp`** and **`remotePort`**. The media server will then have the necessary information to transmit media to the remote party.

If the remote parties media parameters change and must be reset on the media server, issue a **`CreateConnection`** action with the existing **`connectionId`** and the new media parameters specified in **`remoteIp`** and **`remotePort`**.

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

## Metreos.Providers.MediaServer.CreateConnectionConference Action

Create a media server connection and immediately place that connection into a conference, or if the conference does not exist, create a new one.

**Operation**

Synchronous

**Required Parameters**

None.

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| conferenceId | System.Int32 | Unique media server ID for a conference |
| connectionId | System.Int32 | Unique media server ID for a connection |
| remoteIp | System.String | IP address of the remote media endpoint |
| remotePort | System.UInt16 | Port on which the remote media endpoint is listening for media |
| soundToneOnJoin | System.Boolean | Indicates whether tones are played when participants join the conference |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| conferenceId | System.String | conferenceId associated with this action |
| connectionId | System.String | Unique media server ID for a connection |
| ipAddress | System.String | Media server IP address |
| port | System.String | Media server port |
| resultCode | System.String | Result of a media server operation |

**Return Value**

Returns **`success`** if the connection was created and added to a conference; otherwise, `failure` is returned. If the media server fails to create the connection, the **`resultCode`** parameter will hold an integer value indicating the reason why the connection could not be created.

**Remarks**

This action allows the application to create a connection to the media server and at the same time either establish a new conference or add the newly created connection into an existing conference.

This action should *not* be used to perform a half-connect to the media server. If that is necessary use **`CreateConnection`** and then use **`CreateConnectionConference`** after the media information for the remote party is known.

When creating a new conference, specify **`0`** for **`conferenceId`** to obtain a **`conferenceId`**

for the newly created connection in the returned **conferenceId** field. If you specify **0** for **conferenceId**, and provide a valid non-zero **connectionId**, a new conference will be created, and the connection with the specified **connectionId** will be placed in the conference. If both **conferenceId** and **connectionId** are specified, the connection associated with **connectionId** will be placed in the conference associated with **conferenceId**. If you're providing a **connectionId**, you may also specify **remoteIp** and **remotePort** for the remote endpoint.

The returned **connectionId** is a copy of the one that you passed in. The returned **ipAddress** and **port** are the IP address and port reserved for a half-to-full connection, if CreateConnectionConference is used in concert with a half connection.

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

## Metreos.Providers.MediaServer.DeleteConnection Action

Deletes an existing media server connection freeing the resources used by that connection for use by other applications.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| conferenceId | System.Int32 | Unique media server ID for a conference |
| connectionId | System.Int32 | Unique media server ID for a connection |

**Optional Parameters**

None

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| resultCode | System.String | Result of a media server operation |

**Return Value**

Returns **`success`** if the connection was removed; otherwise, **`failure`** is returned. If the media server fails to delete the connection, the **`resultCode`** parameter will hold an integer value indicating the reason why the connection could not be deleted.

**Remarks**

Once an application is finished using a media server connection, it must delete that connection to free all resources that were being used by the connection. If an application creates more than one connection, all of the connections it created must be properly deleted.

Applications may delete connections at any time, regardless of whether an operation such as playing an announcement is currently executing on that media server connection. If an application does delete a connection that is currently executing an asynchronous media server operation then the application will not receive a final asynchronous event for that operation. For example, if an announcement is being played to a connection and the application deletes that connection, **no `PlayAnnouncement_Complete`** will be fired to the application.

MCE requires the application to provide either the **`connectionId`** or the **`conferenceId`** or both.

- To delete a single connection, specify only **`connectionId`**.
- To close an entire conference and delete all connections within it, specify only **`conferenceId`**.
- To remove a connection from a conference without deleting it, specify both **`connectionId`** and **`conferenceId`**.

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

## Metreos.Providers.MediaServer.MuteConferenceConnection Action

Mute an individual connection in a conference.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| conferenceId | System.Int32 | Unique media server ID for a conference |
| connectionId | System.Int32 | Unique media server ID for a connection |

**Optional Parameters**

None

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| resultCode | System.String | Result of a media server operation |

**Return Value**

Returns **`success`** if the connection was muted. Otherwise, **`failure`** is returned. If the media server operation fails the **`resultCod`** parameter will hold an integer value indicating the specific reason for the failure.

**Remarks**

The **`MuteConferenceConnection`** action allows applications to individually mute specific connections within a conference. The media server sets the connection's mode to **receive only** allowing the connection to continue to receive audio, but audio received from that connection is ignored.

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

## Metreos.Providers.MediaServer.PlayAnnouncement Action

Play an audio file announcement to a specific media server connection. The announcement can be played to all connections currently in a conference.

**Operation**

Asynchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application. Metreos recommends leaving this parameter set to the default **none**. |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| audioFileAttribute | System.String | Attributes of the media file (i.e. format, encoding, bitrate) (deprecated) |
| audioFileBitrate | System.Int32 | The bit rate of the media file |
| audioFileEncoding | System.String | The encoding of the media file ('ulaw' or 'alaw') |
| audioFileFormat | System.String | The format of the media file ('vox' or 'wav') |
| conferenceId | System.Int32 | Unique media server ID for a conference |
| connectionId | System.Int32 | Unique media server ID for a connection |
| filename | System.String | Name of the 1st file to play |
| filename2 | System.String | Name of the 2nd file to play |
| filename3 | System.String | Name of the 3rd file to play |
| state | System.String | User data which can be hair-pinned in a media server operation (deprecated) |
| termCondDigit | System.String | Digit to terminate the play announcement on. |
| termCondDigitList | System.String | Digit list to observe before terminating the play announcement command |
| termCondDigitPattern | System.String | A specific digit pattern to observe before terminating the play announcement command. |
| termCondMaxDigits | System.Int32 | Number of digits to receive before terminating the play announcement. |
| termCondMaxTime | System.Int32 | Interval in milliseconds to wait before terminating the play announcement. Interval of silence in milliseconds to observe before terminating the play announcement. |
| termCondNonSilence | System.Int32 | Amount of non-silence in milliseconds to observe before terminating the play announcement. |

| | | |
|---|---|---|
| termCondSilence | System.Int32 | Amount of silence in milliseconds to observe before terminating the command |
| terminationCondition | System.String | Condition under which the operation should complete (deprecated) |

**Result Data Fields**

None.

**Return Value**

Returns `success` if the announcement has begun playing; otherwise, `failure` is returned. If the media server operation fails the `resultCode` parameter will hold an integer value indicating the specific reason for the failure.

Once the announcement starts playing the application will be notified that it is finished playing upon receipt of a `PlayAnnouncement_Complete` event.

**Remarks**

Announcements may be played to either a single connection or an entire conference on the media server. The media server will always stop playing an announcement once the end of the announcement file has been reached; however, the application may modify this behavior by adding additional termination conditions.

Termination conditions may be specified to stop the play announcement if a specific DTMF digit is received, if a certain amount of silence is observed, after a specific period of time, or if a specific number of DTMF digits are received. Some valid termination condition examples are:

- `maxtime 3000`
- `digit #`
- `maxdigits 3`
- `silence 60000`

NOTE: The `terminationCondition` parameter has been deprecated and is only present for backwards compatibility. Use the individual termination condition parameters such as `termCondDigit` for new application development.

The `connectionId` parameter is required only if `conferenceId` is not specified. If both parameters are specified then `connectionId` will take precedence.

The `filename` parameter refers to a file on the media server machine. The value of the parameter should be a relative path and filename to the announcement file to be played.

Playing audio files that are not recorded in VOX format requires the application to specify an `audioFileAttribute` specifying the file type of the announcement file. Currently, the only other supported file format is WAV. To play WAV files, specify `format wav` in the `audioFileAttribute` parameter. However, this is not required if the application is playing back an announcement that was previously recorded by the media server during a `RecordAudio` action. Note, WAV files should be recorded in PCM, 8kHz, 8bit Mono format.

NOTE: The `audioFileAttribute` parameter has been deprecated and is only present for backwards compatibility. Use the individual audio file attribute parameters such as

`audioFileFormat` for new application development.

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

## Metreos.Providers.MediaServer.ReceiveDigits Action

Indicates to the media server that it should begin to monitor the media stream of a particular connection for DTMF digits.

**Operation**

Asynchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| connectionId | System.Int32 | Unique media server ID for a connection |
| terminationCondition | System.String | Condition under which the operation should complete |
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application. Use the default value of 'none' unless you need to change this. |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| termCondDigit | System.String | A digit to observe that will terminate the command |
| termCondDigitList | System.Int32 | A list of digits to observe that will terminate the command |
| termCondDigitPattern | System.Int32 | A specific pattern of digits to observe that will terminate the command |
| termCondMaxDigits | System.Int32 | Number of digits to receive before terminating the command |
| termCondMaxTime | System.Int32 | Amount of time in milliseconds to wait before terminating the command |
| termCondInterDigitDelay | System.Int32 | The maximum amount of time between digits to allow before terminating the receive digits command. |
| terminationCondition | System.String | Condition under which the operation should complete (deprecated) |
| state | System.String | Optional user state information to be returned when asynchronous command completes. |

**Result Data Fields**

None.

**Return Value**

Returns `success` if the media server has begun to watch for DTMF digits; otherwise `failure` is returned.

When the media server stops watching for DTMF digits on the media stream of a connection, a `ReceiveDigits_Complete` event will be fired to the application. If an error occurs while processing the digits a `ReceiveDigits_Failed` event will be fired to the application.

**Remarks**

When receiving digits the media server continuously checks the digits received against the originally specified termination conditions. For example, termination conditions for the `ReceiveDigits` action could be to terminate on a specific number of digits received or when a specific digit (such as `#`) is received.

Termination conditions may be specified to stop the receive digits if a specific DTMF digit is received, if a specific list of DTMF digits is received, after a specific period of time without receiving any DTMF digits, or if a specific number of DTMF digits are received.

NOTE: The `terminationCondition` parameter has been deprecated and is only present for backwards compatibility. Use the individual termination condition parameters such as `termCondDigit` for new application development.

The `ReceiveDigits` action will terminate when one of the specified termination conditions are observed. When this occurs, a `ReceiveDigits_Complete` event will fire to the application containing a parameter with the digits that were received.

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

## Metreos.Providers.MediaServer.RecordAudio Action

Records audio received on a specific media server connection. Optionally, record the mixed audio produced by a conference.

**Operation**

Asynchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application. Leave this defaulted to 'none' unless you need to change this. |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| audioFileAttribute | System.String | Attributes of the media file (i.e. format, encoding, bitrate) |
| audioFileBitrate | System.Int32 | The bit rate of the media file |
| audioFileEncoding | System.String | The encoding of the media file ('ulaw' or 'alaw') |
| audioFileFormat | System.String | The format of the media file ('vox' or 'wav') |
| commandTimeout | System.Int32 | Max time to record (ms) |
| conferenceId | System.Int32 | Unique media server ID for a conference |
| connectionId | System.Int32 | Unique media server ID for a connection |
| expires | System.Int32 | Amount of time in days to keep recorded file on server |
| filename | System.String | Name of the file to create containing the recorded audio |
| state | System.String | User data which can be hair-pinned in a media server operation (deprecated) |
| termCondDigit | System.String | A digit to observe that will terminate the command |
| termCondMaxTime | System.Int32 | Amount of time in milliseconds to wait before terminating the command |
| termCondNonSilence | System.Int32 | Amount of non-silence in milliseconds to observe before terminating the command |
| termCondSilence | System.Int32 | Amount of silence in milliseconds to observe before terminating the command |
| terminationCondition | System.String | Condition under which the operation should complete |

**Result Data Fields**

| Parameter Name | Data Type | Description |
|---|---|---|
| connectionId | System.Int32 | Id of the session |

**Return Value**

Returns **`success`** if the media server has begun to record audio. Otherwise **`failure`** is returned.

When the media server stops recording a **`RecordAudio_Complete`** event will be fired to the application. If an error occurred while recording, a **`RecordAudio_Failed`** event will be fired instead.

**Remarks**

Audio may be recorded from either a single connection or an entire conference on the media server. The media server will record audio until one of the specified termination conditions is satisfied.

Termination conditions may be specified to stop the record audio if

- A specified DTMF digit is received
- A specified interval of silence is observed
- A specified time period
- A specified interval of non-silence is observed

**NOTE**: The **`terminationCondition`** parameter has been deprecated and is only present for backward compatibility. Use the individual termination condition parameters such as `termCondDigit` for new application development.

The application may specify the file to be recorded to. Alternatively, if no `filename` parameter is specified then the media server will automatically generate a random file name. The file name that was recorded will be returned to the application when the final asynchronous event, either **`RecordAudio_Complete`** or **`RecordAudio_Failed`**, is fired to the application.

The application may also specify the format of the recording, encoding of the recorded file, and recording bit rate. Furthermore, the **`expires`** parameter specifies the amount of time in days that the media server should retain the recorded file.

**NOTE**: The **`audioFileAttribute`** parameter has been deprecated and is only present for backward compatibility. Use the individual audio file attribute parameters such as **`audioFileFormat`** for new application development.

The **`connectionId`** parameter is required only if **`conferenceId`** is not specified. If both parameters are specified then `connectionId` will take precedence.

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

## Metreos.Providers.MediaServer.SendDigits Action

Inserts digits into the processing stream of the media server.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| connectionId | System.Int32 | Unique media server ID for a connection |
| digits | System.String | A string of DTMF digits |

**Optional Parameters**

None

**Result Data Fields**

None.

**Return Value**

Returns **`success`** if the digits were successfully inserted. Otherwise, **`failure`** is returned.

**Remarks**

Applications sometimes receive digits from the signaling path rather than media stream. In such cases it is necessary to insert those digits into the media stream to allow the media server to match those digits against existing termination conditions. **`SendDigits`** allows applications to proxy out-of-band DTMF digits back to the media server for processing.

Typically, this action would be used in response to the call control event **`Metreos.Providers.H323.GotDigits`**. **Metreos.Providers.H323.GotDigits** fires when out-of-band digits are received. After receiving the **Metreos.Providers.H323.GotDigits** event invoke the **`SendDigits`** action and set the `digits` parameter to the value of the parameter with the same name from the **`Metreos.Providers.H323.GotDigits`** event.

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

## Metreos.Providers.MediaServer.DetectSilence Action

Inserts digits into the processing stream of the media server.

**Operation**

Asynchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| connectionId | System.Int32 | Unique media server ID for a connection |
| silenceTime | System.Int32 | Amount of silence to observe |
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application |

**Optional Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| commandTimeout | System.String | Amount of time before silence detection stops |
| state | System.String | User state information to be returned when asynchronous command completes |

**Result Data Fields**

None.

**Return Value**

Returns `success` if the media server was able to begin detecting silence; otherwise, **`failure`** is returned.

When the media server detects silence, a **`DetectSilence_Complete`** event will be fired to the application. If the command timeout period elapses first, **a `DetectSilence_Failed`** event will be fired instead.

**Remarks**

If **`commandTimeout`** is not specified, the value in the **`mmsconfig.properties`** file will be used instead.

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

## Metreos.Providers.MediaServer.StopMediaOperation Action

Stop an executing, asynchronous media server operation on a specific connection to the media server.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| connectionId | System.Int32 | Unique media server ID for a connection |

**Optional Parameters**

None

**Result Data Fields**

None.

**Return Value**

Returns **`success`** if the operation was stopped. Otherwise, **`failure`** is returned.

**Remarks**

If a media server connection is currently executing an asynchronous media server operation such as **`PlayAnnouncement`** the application may interrupt that operation. To do so issue the **`StopMediaOperation`**. When interrupted a final asynchronous event will be fired to the application. The asynchronous event that is fired will contain a **`reason`** parameter whose value will be **`userstop`** to indicate that the command was terminated early at the request of the user.

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

## Metreos.Providers.MediaServer.UnMuteConferenceConnection Action

Un-mutes a previously muted connection in a conference.

**Operation**

Synchronous

**Required Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| conferenceId | System.Int32 | Unique media server ID for a conference |
| connectionId | System.Int32 | Unique media server ID for a connection |

**Optional Parameters**

None

**Result Data Fields**

None.

**Return Value**

Returns **`success`** if the connection was un-muted. Otherwise, **`failure`** is returned.

**Remarks**

The **`UnMuteConferenceConnection`** action allows applications to individually un-mute specific connections within a conference. The media server sets the mode of the connection to **send and receive** allowing the connection to continue to receive audio and allowing audio received from that connection to be mixed into the conference.

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

## Asynchronous Callback Events

### Metreos.Providers.MediaServer.PlayAnnouncement_Complete Event

Event fired indicating successful completion of a **`PlayAnnouncement`** action.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| resultCode | System.Int32 | Result of a media server operation |
| state | System.String | User data which can be hair-pinned in a media server operation (deprecated) |
| terminationCondition | System.String | Condition under which the operation completed |
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application |

**Remarks**

Upon receiving this event, the **`terminationCondition`** parameter contains the reason that the media server stopped playing audio. The **`terminationCondition`** parameter can be one of the following:

- **`digit`**
- **`eod`**
- **`maxdigits`**
- **`maxtime`**
- **`silence`**
- **`userstop`**

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

# Metreos.Providers.MediaServer.PlayAnnouncement_Failed Event

Event fired indicating failure of a **`PlayAnnouncement`** action.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| resultCode | System.Int32 | Result of a media server operation |
| state | System.String | User data which can be hair-pinned in a media server operation (deprecated) |
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application |

**Remarks**

An error occurred, causing the **`PlayAnnouncement`** action to terminate prematurely without matching a specific termination condition.

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

## Metreos.Providers.MediaServer.ReceiveDigits_Complete Event

Event fired indicating successful completion of a **`ReceiveDigits`** action.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| state | System.String | User data which can be hair-pinned in a media server operation (deprecated) |
| digits | System.String | A string of DTMF digits |
| resultCode | System.Int32 | Result of a media server operation |
| terminationCondition | System.String | Condition under which the operation completed |
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application |

**Remarks**

Upon receiving this event the `digits` parameter will contain all digits received by the media server for the given connection ID. The **`terminationCondition`** parameter contains the reason that the media server stopped receiving digits. The `terminationCondition` parameter can be one of the following:

- **`digit`**
- **`digitlist`**
- **`digitpattern`**
- **`eod`**
- **`maxdigits`**
- **`maxtime`**
- **`userstop`**

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

## Metreos.Providers.MediaServer.ReceiveDigits_Failed Event

Event fired indicating failure of a **`ReceiveDigits`** action.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| state | System.String | User data which can be hair-pinned in a media server operation (deprecated) |
| resultCode | System.Int32 | Result of a media server operation |
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application |

**Remarks**

An error occurred which caused the **`ReceiveDigits`** action to terminate prematurely without matching a specific termination condition.

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

# Metreos.Providers.MediaServer.RecordAudio_Complete Event

Event fired indicating successful completion of a **`RecordAudio`** action.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| filename | System.String | Name of the file created containing the recorded audio |
| resultCode | System.Int32 | Result of a media server operation |
| terminationCondition | System.String | Condition under which the operation completed |
| state | System.String | User data which can be hair-pinned in a media server operation (deprecated) |
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application |

**Remarks**

Upon receiving this event the `filename` parameter will contain the name of the file that the audio was recorded to. The `terminationCondition` parameter contains the reason that the media server stopped recording audio. The **`terminationCondition`** parameter can be one of the following:

- **`digit`**
- **`eod`**
- **`maxtime`**
- **`nonsilence`**
- **`silence`**
- **`userstop`**

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

# Metreos.Providers.MediaServer.RecordAudio_Failed Event

Event fired indicating failure of a `RecordAudio` action.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| resultCode | System.Int32 | Result of a media server operation |
| state | System.String | User data which can be hair-pinned in a media server operation (deprecated) |
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application |

**Remarks**

An error occurred which caused the **`RecordAudio`** action to terminate prematurely without matching a specific termination condition.

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

# Metreos.Providers.MediaServer.DetectSilence_Complete Event

Event fired indicating successful completion of a **`DetectSilence`** action.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| resultCode | System.Int32 | Result of a media server operation |
| state | System.String | User data which can be hair-pinned in a media server operation (deprecated) |
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

# Metreos.Providers.MediaServer.DetectSilence_Failed Event

Event fired indicating failure of a **`DetectSilence`** action.

**Type**

Asynchronous Callback

**Event Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| resultCode | System.Int32 | Result of a media server operation |
| state | System.String | User data which can be hair-pinned in a media server operation (deprecated) |
| userData | System.String | An opaque token used to correlate asynchronous actions with final responses in an application |

**Remarks**

**Requirements**

Metreos Framework version 2.0

Metreos Media Engine Provider version 2.0

## 2.16. Standard Types

Following is a list of types which should for all purposes be treated the same as their .NET counterpart:

| Type Name |
|---|
| **Metreos.Types.ArrayList** |
| **Metreos.Types.Bool** |
| **Metreos.Types.DataSet** |
| **Metreos.Types.DataTable** |
| **Metreos.Types.DateTime** |
| **Metreos.Types.Double** |
| **Metreos.Types.Hashtable** |
| **Metreos.Types.Int** |
| **Metreos.Types.Long** |
| **Metreos.Types.Queue** |
| **Metreos.Types.Short** |
| **Metreos.Types.SortedList** |
| **Metreos.Types.Stack** |
| **Metreos.Types.String** |
| **Metreos.Types.StringCollection** |
| **Metreos.Types.StringDictionary** |
| **Metreos.Types.UInt** |
| **Metreos.Types.ULong** |
| **Metreos.Types.UShort** |

# 3. APPENDIX B: ATTRIBUTES

## Metreos.PackageGeneratorCore.ActionAttribute

Attribute used to declare an action handler.

**Formats**

| Use to declare | Usage |
|---|---|
| Native Actions | `[Action(nativeActionName, allowCustomParams, displayName, description)]` |
| Provider Actions | `[Action(providerActionName, allowCustomParams, displayName, description, async)]` |
| Provider Actions with unsolicited callbacks | `[Action(providerActionName, allowCustomParams, displayName, description, async, unsolicitedCallbacks)]` |

**Attribute Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| nativeActionName | System.String | The name of the native action |
| providerActionName | System.String | The name of the provider action |
| allowCustomParams | System.Boolean | Boolean specifying whether the action takes custom parameters |
| displayName | System.String | The name that will appear for this action inside Max Visual Designer |
| description | System.String | The description associated with the action |
| async | System.Boolean | Boolean specifying whether this action is asynchronous or not |
| unsolicitedCallbacks | System.String[] | An array of strings. Contains the exact names of the unsolicited events this action may cause |

**Remarks**

This attribute is used to declare the behavior of an action's **Execute** method.
When you specify unsolicited callbacks for a provider action, you can expect the specified events to occur.

**Requirements**

Metreos Framework version 2.0

# Metreos.PackageGeneratorCore.ActionParamFieldAttribute

Attribute used to declare an action parameter

**Formats**

| Use to declare | Usage |
|---|---|
| Action Parameters | [ActionParamField()] |
| Action Parameters | [ActionParamField(description)] |
| Action Parameters | [ActionParamField(mandatory)] |
| Action Parameters | [ActionParamField(description, mandatory)] |

**Attribute Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| description | System.String | The description of the parameter |
| mandatory | System.Boolean | If **mandatory** is **true**, this is a required parameter, otherwise it is optional |

**Remarks**

This attribute is used to declare an action parameter field. [ActionParamField()] sets **description** to null and **mandatory** to true.

**Requirements**

Metreos Framework version 2.0

# Metreos.PackageGeneratorCore.PackageDeclAttribute

Attribute used to declare a package

**Formats**

| Use to declare | Usage |
|---|---|
| Packages | `[PackageDecl(_namespace)]` |
| Packages | `[PackageDecl(_namespace, description)]` |

**Attribute Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| _namespace | System.String | The namespace of the package |
| description | System.Type | The description of this package |

**Remarks**

This attribute is used to declare the namespace in which the action will reside. Multiple related actions can be grouped into the same package by specifying the same namespace. User-defined namespaces should use the following format:

*company_name*.**Native**.*category_name.action_name*

**Requirements**

Metreos Framework version 2.0

# Metreos.PackageGeneratorCore.ResultDataFieldAttribute

Attribute used to declare a result parameter.

**Formats**

| Use to declare | Usage |
|---|---|
| Result Parameters | `[ResultDataField()]` |
| Result Parameters | `[ResultDataField(description)]` |

**Attribute Parameters**

| Name | Data Type | Description |
|---|---|---|
| description | System.String | The description of this parameter |

**Remarks**

This attribute is used to declare result parameter fields. [ResultDataField()] sets **description** to null.

**Requirements**

Metreos Framework version 2.0

# Metreos.PackageGeneratorCore.ResultValueAttribute

Attribute used to declare a result parameter.

**Formats**

| Use to declare | Usage |
|---|---|
| Result Value | `[ReturnValue()]` |
| Result Value | `[ReturnValue(type, description)]` |
| Result Value | `[ReturnValue(customEnumType, description)]` |

**Attribute Parameters**

| Name | Data Type | Description |
|---|---|---|
| customEnumType | System.Type | A user defined enum containing possible return values for this action |
| type | Metreos. PackageGeneratorCore. PackageXml. returnValueTypeType | A Metreos pre-defined return value. |
| description | System.String | The description of this value |

**Remarks**

This attribute is used to declare result values. These are values such as **success** and **failure** that are used by an application to branch. **customEnumType** is a user-defined enum that contains the values on which a user wants the application to branch. **type** is an enum of pre-defined return values. Possible values for **type** are

- **Metreos.PackageGeneratorCore.PackageXml.success**
- **Metreos.PackageGeneratorCore.PackageXml.boolean**
- **Metreos.PackageGeneratorCore.PackageXml.yes_no**
- **Metreos.PackageGeneratorCore.PackageXml.custom**

**Requirements**

Metreos Framework version 2.0

# Metreos.PackageGeneratorCore.TypeInputAttribute

Attribute used to declare a type that can be assigned to a native type.

**Formats**

| Use to declare | Usage |
|---|---|
| Type of input | `[TypeInputAttribute(type, description)]` |

**Attribute Parameters**

| Name | Data Type | Description |
|---|---|---|
| description | System.String | Description of the **type** |
| type | System.String | The type of the variable |

**Remarks**

This attribute is used to describe what types a Native Type variable can hold.
**type** should be a string representation of a type for example **System.String**

**Requirements**

Metreos Framework version 2.0