

Cisco TelePresence IP VCR API 2.7

Product Programming Reference Guide

D14661.04

March 2011

Contents

Introduction	4
API History	4
XML-RPC implementation	4
Transport protocol	4
API overhead	5
API overview	6
Encoding	6
Specify encoding with HTTP headers	6
Specify encoding with XML header	6
Authentication	6
Message flow	6
Enumerate methods	8
Enumerate filters	8
Binary operators	9
Unary operators	9
Revision numbers	10
Using revision numbers with enumerate methods	10
Discovering record removal	11
Dead records	11
API reference	12
addressBookEntry.enumerate	12
connection.enumerate	17
connection.streaming.enumerate	22
connection.streaming.terminate	26
connection.terminate	27
device.disk.query	27
device.health.query	27
device.network.query	30
device.query	33

device.restartlog.query.....	34
folder.enumerate.....	35
gatekeeper.query.....	38
gateway.enumerate.....	40
recording.callout.....	42
recording.configure.....	43
recording.delete.....	44
recording.enumerate.....	44
recording.play.....	48
recording.stop.....	49
sip.query.....	49
Feedback receivers.....	51
feedbackReceiver.configure.....	51
feedbackReceiver.query.....	52
Feedback messages.....	53
Feedback events.....	53
Related information.....	55
Fault codes.....	55
HTTP keep-alives.....	56
system.xml file.....	57
Checking for updates and getting help.....	58
References.....	59

Introduction

This document accompanies the latest version of the remote management API for the Cisco TelePresence IP VCR software (respectively referred to as API and IP VCR in this document). The following Cisco TelePresence products support this API when they are running IP VCR version 3.0 and later:

- Cisco TelePresence IP VCR 2200 series (models IP VCR 2210, IP VCR 2220, IP VCR 2240)
- Cisco TelePresence VCR MSE 8220 (blade that fits in the MSE 8000 chassis)

API History

The following table shows the device's software versions and the corresponding supported API versions:

API version	IP VCR version
2.7 (this version)	3.0 and later
2.5	2.3 and later
2.4	2.2

XML-RPC implementation

API calls and responses are implemented using the XML-RPC protocol. This simple protocol does remote procedure calling using HTTP as the transport and XML as the encoding. It is extremely simple although it does still allow for complex data structures. XML-RPC is not platform-dependent and was chosen in favor of SOAP (Simple Object Access Protocol) because of its simplicity.

The interface is stateless, which means the application must either regularly poll the device for status or continually listen to the device - if it is configured to publish feedback events.

The API implements all parameters and returned data as `<struct>` elements, each of which is explicitly named. For example, `device.query` returns (amongst other data) the current time as:

```
<member>
  <name>currentTime</name>
  <value><dateTime.iso8601>20110121T13:31:26</dateTime.iso8601></value>
</member>
```

rather than simply

```
<dateTime.iso8601>20110121T13:31:26</dateTime.iso8601>
```

Note: Unless otherwise stated, assume strings have a maximum length of 32 characters.

Refer to the [XML-RPC specification^{\[1\]}](#) for more information.

Transport protocol

The device implements HTTP/1.1 as defined by [RFC 2616^{\[2\]}](#). It expects to receive HTTP communications over TCP/IP connections to port 80. The application should send HTTP POST messages to the URL `/RPC2` on the device's IP address.

HTTPS is supported on all IP VCR products from IP VCR software version 2.3 and later.

HTTPS is provided on TCP port 443 by default, although you can configure the device to receive HTTP and HTTPS on non-standard TCP port numbers if necessary.

API overhead

Every API command that your application sends incurs a processing overhead within the target device's own application. The amount of overhead varies with the type of command and parameters. A high API processing load may impair the device's performance – in the same way as if several users simultaneously accessed the device's web interface. Bear this in mind when you design your application's architecture and software.

For this reason, we recommend that you use a single server to run the calling application and send commands to the device. If users need concurrently use the application, provide a web interface on your application server or write a client to communicate with that server. The server then manages the client requests and only the server sends API commands directly to the device.

We also recommend that you implement some control in your application to prevent the device being overloaded with API commands.

These design considerations provide more control than allowing clients to send API commands directly and will prevent the device's performance from being impaired by unmanageable API load.

API overview

Encoding

Your application can encode messages as ASCII text or as UTF-8 Unicode. If you do not specify the encoding, the API assumes ASCII encoding. You can specify the encoding in a number of ways:

Specify encoding with HTTP headers

There are two ways of specifying UTF-8 in the HTTP headers:

- Use the **Accept-Encoding: utf-8** header
- Modify the **Content-Type** header to read **Content-Type: text/xml; charset=utf-8**

Specify encoding with XML header

The `<?xml>` tag is required at the top of each XML file. The API will accept an encoding attribute for this tag; that is, `<?xml version="1.0" encoding="UTF-8"?>`.

Authentication

The controlling application must authenticate itself on the device as a user with administrative privileges. Also, because the interface is stateless, every call must contain authentication parameters:

`authenticationUser`

Type: **string**

Name of a user with sufficient privilege for the operation being performed. The name is case sensitive.

`authenticationPassword`

Type: **string**

The password that corresponds with the given `authenticationUser`. The API ignores this parameter if the user has no password. This behavior differs from the web interface, where a blank password must be blank.

Note: Authentication information is sent using plain text and should only be sent over a trusted network.

Message flow

The application initiates the communication and sends a correctly formatted XML-RPC command to the device.

Example command

```
<?xml version='1.0' encoding='UTF-8'?>
  <methodCall>
    <methodName>recording.delete</methodName>
```

```

<params>
  <param>
    <value>
      <struct>
        <member>
          <name>authenticationPassword</name>
          <value><string></string></value>
        </member>
        <member>
          <name>recordingId</name>
          <value><int>101</int></value>
        </member>
        <member>
          <name>authenticationUser</name>
          <value><string>admin</string></value>
        </member>
      </struct>
    </value>
  </param>
</params>
</methodCall>

```

Assuming the command was well formed, and that the device is responsive, the device will respond in one of these ways:

- With an XML **methodResponse** message that may or may not contain data, depending on the command.
- With an XML **methodResponse** that includes only a fault code message.

Example success

```

<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>status</name>
            <value>
              <string>operation successful</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>

```

Example fault code

```

<?xml version="1.0"?>

```

```

<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value>
            <int>22</int>
          </value>
        </member>
        <member>
          <name>faultString</name>
          <value>
            <string>no such recording</string>
          </value>
        </member>
      </struct>
    </value>
  </fault>
</methodResponse>

```

Enumerate methods

Enumerate methods have the potential to return a large volume of data, so these calls have a control mechanism to limit the number of enumerated items per call.

Each enumerate call may take and return an **enumerateID** parameter which tells the API or calling application where to start the enumeration. The mechanism works as follows:

1. The application calls an enumerate method without an **enumerateID** parameter.
2. The device returns an array containing the enumerated items, and possibly an **enumerateID**. The response will always include an **enumerateID** if the device enumerated more items than it included in the response.
3. If there is an **enumerateID**, the application should call the enumerate method again, supplying the **enumerateID** as returned by the previous call.
4. The application should repeat this process until the response fails to include an **enumerateID**. This means that the enumeration is complete.

Note: Do not supply your own **enumerateID** values; make sure you only use the values returned by the device.

Enumerate filters

Enumerate methods will accept an optional **enumerateFilter** parameter, which allows you to filter the response. The parameter must contain a filter expression, which is built from criteria and operators.

The filter criteria that a call will accept vary depending on the call, but the syntax for using those criteria in expressions is the same for all methods that allow filtering. The reference information for methods that allow filtering includes acceptable filter criteria.

If the filter expression evaluates to true for the enumerated item, the item will be included in the device's response. If the expression evaluates false, the enumerated item will be filtered out of the response.

Filter expressions consist of atomic expressions combined with operators and parentheses. Whitespace is ignored. Functions are valid, and any parameters are in a comma separated list in parentheses after the function name, for example, `function(expression1, expression2)`.

For example, if the expression `(inProgress && internal)` is used to filter the response to `recording.enumerate`, the returned array of recordings will only include those which are both `inProgress` and `internal`.

The integer 0 evaluates to false and all other integers to true. Integers can be expressed using any string of valid digits. Prefix hex digits with `0x`, decimal with `0t` and binary with `0z`. The API assumes decimal if you don't supply a prefix.

Binary operators

The following binary operators are valid, in order of priority (lowest priority first):

Operator	Description
<code> </code>	Boolean or
<code>&&</code>	Boolean and
<code> </code>	Bitwise or
<code>^</code>	Bitwise exclusive or
<code>&</code>	Bitwise and
<code>==</code>	Equality
<code>!=</code>	Inequality
<code><</code>	Less than
<code><=</code>	Less than or equal
<code>>=</code>	Greater than or equal
<code>></code>	Greater than
<code><<</code>	Bitwise left shift
<code>>></code>	Bitwise right shift
<code>+</code>	Addition
<code>-</code>	Subtraction
<code>*</code>	Multiplication
<code>/</code>	Division
<code>%</code>	Modulo

Unary operators

The following unary operators are valid. All of these bind tighter than any binary operator.

Operator	Description
<code>-</code>	Unary minus

+	Unary plus
!	Logical negation
~	Bitwise negation

Revision numbers

Note: This feature is available from API version 2.7 onwards. Your application can determine the API version supported by the device by reading the value of `apiVersion` returned by `device.query`.

To reduce the size of responses when querying the device, some of the enumeration methods support a revision number system.

When the device responds to a call that supports revision numbers, it returns an extra integer field called `currentRevision`. For example:

```
<member>
  <name>currentRevision</name>
  <value>
    <int>18</int>
  </value>
</member>
```

The revision number increases every time any API query is made on the device. To reduce the size of subsequent query responses, you may pass in the `lastRevision` parameter. For example:

```
<member>
  <name>lastRevision</name>
  <value>
    <int>18</int>
  </value>
</member>
```

The device returns only those records that have changed since `lastRevision`. For example, if you provide a `lastRevision` parameter in a `connection.enumerate` call, the device's enumeration response only includes connections that changed since its revision number was set to the value you provided.

Using revision numbers with enumerate methods

When you use revision numbers with enumerate methods, you should use the same value of the `lastRevision` parameter for each stage of the enumeration, *despite that a newer `currentRevision` parameter is returned at each stage*. If you update `lastRevision` to use the newer `currentRevision`, the device will not return the rest of the changes you were interested in; it will only look for changes since you started the enumeration.

Similarly, if you want to store a new value to use as `lastRevision` in a future enumeration, you should use the `currentRevision` number that the device returned in the first response to your current enumeration.

You need to use the same revision number throughout enumerations to ensure that the device reports all records that have changed, but this means that occasionally a record is reported more than once when it has only changed once.

Discovering record removal

The problem with the revision number feature only returning changed records is that the calling application can't tell whether a record has been *removed altogether*.

One approach to solving this problem is the `listAll` parameter, which a client application may set to `true` to tell the device to return every record available. This allows the client to synchronize with the device because it can safely assume that any record not returned by this request (or series of requests, in the case of enumerations) no longer exists on the device.

For example, you can assume that any connections not returned by `connection.enumerate` when `listAll` is set to `true` have been removed from the device.

You can use the `listAll` parameter in conjunction with the `lastRevision` parameter. In this case, the device returns every record it has but may remove data from members whose records have not changed since `lastRevision`. The API inserts a parameter named `changed` instead, with its value set to `false`; the calling application can ignore those members because they haven't changed since `lastRevision`, and the response is still much smaller than it would otherwise be with `listAll`.

Dead records

Another approach to the record removal problem is the `dead` parameter. The device maintains a cache of records that have been removed and are no longer considered active in any sense. It will return the `dead` parameter, with value `true`, instead of those records if those records would otherwise have been required by the response.

The device will never return a dead record unless revision numbers are being used. The device will also never return a dead record if `listAll` is set to `true`.

Furthermore, dead records are only cached for a few minutes.

The device only returns a dead record under the following conditions:

- `listAll` is not set, or is set `false`
- The call supports revision numbers and `lastRevision` is supplied
- The record was removed at some point after the supplied `lastRevision`
- That record has not yet been cleared from the cache.

When these conditions are met, the query response includes the minimum of information required to identify the record as well as the `dead` parameter, set to `true`. The calling application can safely assume that the device will soon remove any trace of this record.

However, unless the client is doing frequent, regular polling, we recommend using the `listAll` parameter, as described above, to verify removed records.

API reference

This is a list of the API calls supported by the target device. For each API call, the following information is provided where applicable:

- Description of the call's function and status
- Accepted parameters
- Returned parameters, structure formats and data types
- Deprecated parameters

Click the call name to read a detailed description of the call.

- [addressBookEntry.enumerate](#)
- [connection.enumerate](#)
- [connection.streaming.enumerate](#)
- [connection.streaming.terminate](#)
- [connection.terminate](#)
- [device.disk.query](#)
- [device.health.query](#)
- [device.network.query](#)
- [device.query](#)
- [device.restartlog.query](#)
- [folder.enumerate](#)
- [gatekeeper.query](#)
- [gateway.enumerate](#)
- [recording.callout](#)
- [recording.configure](#)
- [recording.delete](#)
- [recording.enumerate](#)
- [recording.play](#)
- [recording.stop](#)
- [sip.query](#)

addressBookEntry.enumerate

Status: **active**

Enumerates the configured endpoints on this device.

Accepts:

enumerateID

Type: **string**

Enumerate calls may return many results so all of them will accept this parameter and may include this parameter in the response.

If the response includes an enumerateID, the application should pass the ID to the subsequent enumerate call to retrieve the next set of responses. If the response does not include an enumerateID, there are no more results in the enumeration.

If the application omits the enumerateID, the target device will start a new enumeration and return the first set of results.

Returns:

The next `enumerateID` up from the one you provided, and the associated addressbook entries.

`enumerateID`

Type: **string**

Enumerate calls may return many results so all of them will accept this parameter and may include this parameter in the response.

If the response includes an `enumerateID`, the application should pass the ID to the subsequent `enumerate` call to retrieve the next set of responses. If the response does not include an `enumerateID`, there are no more results in the enumeration.

If the application omits the `enumerateID`, the target device will start a new enumeration and return the first set of results.

`addressBookEntries`

Type: **array**

See below for details.

Addressbook entry structure

`name`

Type: **string**

The name of the enumerated item, e.g. gateway or endpoint.

`address`

Type: **string**

The address of the item, e.g. endpoint or gateway; may be hostname, IP address or e164 number.

`protocol`

Type: **string**

Value	Description
h323	The call uses the H.323 protocol
sip	The call uses the SIP protocol

`gatewayName`

Type: **string**

Present in entries for H.323 endpoints which are configured to use a gateway. This name corresponds to the `name` parameter of a gateway returned by `gateway.enumerate`.

`gatewayAddress`

Type: **string**

The address of an H.323 gateway, if required. Only used if protocol is **h323**. This corresponds to the **address** parameter of the gateway as returned by **gateway.enumerate**.

useSIPRegistrar

Type: **boolean**

Not valid unless the protocol is SIP.

Value	Description
true	The endpoint uses the SIP registrar
false	The endpoint does not use the SIP registrar (default)

callInParams

Type: **array**

A structure containing the call in parameters of the endpoint.

callInParams structure

name

Type: **string**

The name of the enumerated item, e.g. gateway or endpoint.

address

Type: **string**

The address of the item, e.g. endpoint or gateway; may be hostname, IP address or e164 number.

e164

Type: **string**

An E.164 number.

conferencingParameters

Type: **struct**

A structure containing the conferencing parameters of the enumerated item, e.g. gateway or endpoint.

conferencingParameters structure

useDefaultMotionSharpness

Type: **boolean**

Value	Description
true	This endpoint will use box-wide default motion sharpness settings.
false	This endpoint will not use box-wide default motion sharpness settings.

minFrameRateMotionSharpness

Type: **integer**

Specifies the minimum frame rate for this endpoint. This parameter is only present if **useDefaultMotionSharpness** is **false**.

useDefaultVideoTransmitResolutions

Type: **boolean**

Value	Description
true	This endpoint will use box-wide default video transmit resolutions
false	This endpoint will not use box-wide default video transmit resolutions

videoTransmitResolutions

Type: **string**

Defines the video resolution that the IP VCR will transmit to this endpoint. The default is to use the box-wide setting, but you can set to one of the following overrides if necessary.

Value	Description
allowAll	The IP VCR may transmit any of the available resolutions to the endpoint.
4to3Only	The IP VCR may only transmit 4:3 video to this endpoint.
4to3WidescreenOverride	The IP VCR may transmit 4:3 video, modified to fit widescreen, to this endpoint.
16to9Only	The IP VCR may only transmit 16:9 video to this endpoint.

maxMediaTxBitRate

Type: **integer**

The maximum media transmission speed from this device, in kbps.

maxMediaRxBitRate

Type: **integer**

The maximum media reception speed of this device, in kbps.

layoutControlDefault

Type: **boolean**

Value	Description
true	The endpoint will use the default (box-wide) layout control settings.
false	The endpoint will not use the default (box-wide) layout control settings.

layoutControlEnabled

Type: **boolean**

Indicates whether the endpoint's participant will have control over their layout. Only present if **layoutControlDefault** is **false**.

Value	Description
true	The participant may change the layout on their endpoint.
false	The participant may not change the layout on their endpoint.

h239ContributionDefault

Type: **boolean**

Selects whether the endpoint will use the box-wide H.239 contribution settings.

Value	Description
true	The endpoint will use the default (box-wide) H.239 contribution settings
false	The endpoint will not use the default H.239 contribution settings.

h239ContributionEnabled

Type: **boolean**

Specifies whether the endpoint will be able contribute H.239. Only present if **h239ContributionDefault** is **false**.

Value	Description
true	The endpoint will be able to participate using H.239
false	The endpoint will not be able to participate using H.239

initialAudioMuted

Type: **boolean**

Value	Description
true	The endpoint's audio is initially muted.
false	The endpoint's audio is not initially muted.

initialVideoMuted

Type: **boolean**

Value	Description
true	The endpoint's video is initially muted.
false	The endpoint's video is not muted.

autoDisconnect

Type: **boolean**

Value	Description
true	Allows the device to automatically disconnect this endpoint, and all remaining endpoints that have this property, when none of the remaining endpoints require manual disconnection.
false	This endpoint requires manual disconnection.

borderWidth

Type: **integer**

Controls the width of the outer border of a preconfigured participant's layout.

Value	Description
0	No border
1	Corresponds to border +1 on the web interface
2	Corresponds to border +2 on the web interface
3	Corresponds to border +3 on the web interface

connection.enumerate

Status: **active**

This call enumerates the active connections to the device.

Accepts:

enumerateFilter

Type: **string**

A filter expression. The enumeration results depend on the supplied expression.

lastRevision

Type: **integer**

This number identifies an earlier set of enumeration data to compare with your current call.

If you supply this parameter using the `currentRevision` value that was returned by a previous enumeration, the current `enumerate` call returns only the differences since that previous call.

If you don't supply this parameter, the device assumes that you want a full enumeration.

`listAll`

Type: **boolean**

Set this to `true` to do a full enumeration, or `false` to return only a subset (use with `lastRevision` parameter).

Filters on:

For the type parameter, you can simply supply the type string's value to the filter expression, for example (`playback && recordingId==5`) which returns all connections watching playback of recording number 5.

`type`

Type: **string**

The connection type.

Value	Description
autoAttendant	This connection is used by an auto attendant.
playback	This connection is used by a playback participant.
recording	This connection is used by a recording participant.

`connectionId`

Type: **integer**

Unique identifier of the connection.

`recordingId`

Type: **integer**

Unique identifier of the recording.

Returns:

`currentRevision`

Type: **integer**

A number that indicates the current revision of this enumeration. You can use this as a `lastRevision` input to a future `enumerate` call to retrieve only the changes between the two enumerations.

`connections`

Type: **array**

An array of active connections. Each member of the array contains information about an active connection to the queried device.

connections array

connectionId

Type: **integer**

Unique identifier of the connection.

type

Type: **string**

The connection type.

Value	Description
autoAttendant	This connection is used by an auto attendant.
playback	This connection is used by a playback participant.
recording	This connection is used by a recording participant.

startTime

Type: **dateTime.iso8601**

Start time of the connection.

Value	Description
20110106T14:00:48	yyyymmddThh:mm:ss

callDuration

Type: **integer**

The duration of the call in seconds.

recordingId

Type: **integer**

Unique identifier of the recording.

recordingName

Type: **string**

The name of the recording. Will be an automatically generated name if no name has been supplied.

state

Type: **string**

The state of the connection. Some states are not applicable to some connection types.

Value	Description
pending	The connection is pending recording, playback, or streaming playback.
active	The connection is currently recording, playing back, or streaming playback.
paused	The playback or streaming playback on this connection is currently paused.
finished	The recording or playback on this connection is finished. This state doesn't apply to streaming connections.
fastForward	The playback on this connection is currently being fast forwarded. This state doesn't apply to streaming connections.
rewind	The playback on this connection is currently being rewound. This state doesn't apply to streaming connections.

playbackPosition

Type: **integer**

Playback position in seconds

mediaDuration

Type: **integer**

The duration in seconds of this playback or recording.

mediaSize

Type: **integer**

The size in kilobytes of the recording. This parameter is only returned for recording connections and is not returned for other connection types.

unicastViewers

Type: **integer**

The count of unicast streaming viewers of this connection. This parameter is only returned for recording connections and is not returned for other connection types. This field is not returned if the recording is in HD.

multicastViewers

Type: **integer**

Number of multicast streaming viewers on a recording connection. This parameter is only returned for recording connections and is not returned for other connection types.

participants

Type: **array**

An array of participants in this recording connection. There are two entries if it's a point-to-point recording but only one otherwise. Each participant member contains the configuration details of the participant.

participants array

displayName

Type: **string**

The display name of the participant.

address

Type: **string**

The address of the item, e.g. endpoint or gateway; may be hostname, IP address or e164 number.

ipAddress

Type: **string**

IPv4 IP address in dotted-quad format.

callIdentifier

Type: **string**

The H.323 ID of the participant.

protocol

Type: **string**

Value	Description
h323	The call uses the H.323 protocol
sip	The call uses the SIP protocol

incoming

Type: **boolean**

Value	Description
true	The call is incoming to the device.
false	The call is not incoming to the device.

maxBitRateToVCR

Type: **integer**

Maximum bit rate to VCR in kbps.

maxBitRateFromVCR

Type: **integer**

The maximum bit rate from the VCR in kbps.

audioRxCodec

Type: **string**

Receive audio codec.

audioTxCodec

Type: **string**

The codec used on the audio transmission.

videoRxCodec

Type: **string**

The receive video codec for this connection.

videoTxCodec

Type: **string**

The transmit video codec for this connection.

contentRxCodec

Type: **string**

The codec used on the incoming content stream.

contentTxCodec

Type: **string**

The codec used to transmit content, if content is being transcoded.

mediaEncryption

Type: **string**

Value	Description
encrypted	The participant's connection is encrypted.
unencrypted	The participant's connection is unencrypted.
rxEncrypted	The participant's incoming connection is encrypted.
txEncrypted	The participant's outbound connection is encrypted.

connection.streaming.enumerate

Status: **active**

This call returns an array of streaming playback connections from the device.

Accepts:**enumerateFilter**Type: **string**

A filter expression. The enumeration results depend on the supplied expression.

lastRevisionType: **integer**

This number identifies an earlier set of enumeration data to compare with your current call.

If you supply this parameter using the `currentRevision` value that was returned by a previous enumeration, the current `enumerate` call returns only the differences since that previous call.

If you don't supply this parameter, the device assumes that you want a full enumeration.

listAllType: **boolean**Set this to `true` to do a full enumeration, or `false` to return only a subset (use with `lastRevision` parameter).**Filters on:****protocol**Type: **string**

The streaming protocol used on this connection.

Value	Description
rtsp	Connection uses the Real Time Streaming Protocol (RTSP)
mms	Connection uses the Microsoft Media Services protocol (MMS)
content	

connectionIdType: **integer**

Unique identifier of the connection.

recordingIdType: **integer**

Unique identifier of the recording.

Returns:**currentRevision**

Type: **integer**

A number that indicates the current revision of this enumeration. You can use this as a **lastRevision** input to a future **enumerate** call to retrieve only the changes between the two enumerations.

connections

Type: **array**

An array of active connections. Each member of the array contains information about an active connection to the queried device.

connections array members

connectionId

Type: **integer**

Unique identifier of the connection.

protocol

Type: **string**

The streaming protocol used on this connection.

Value	Description
rtsp	Connection uses the Real Time Streaming Protocol (RTSP)
mms	Connection uses the Microsoft Media Services protocol (MMS)
content	

startTime

Type: **dateTime.iso8601**

Start time of the connection.

Value	Description
20110106T14:00:48	yyyymmddThh:mm:ss

streamingDuration

Type: **integer**

Duration in seconds of the streamed playback.

recordingId

Type: **integer**

Unique identifier of the recording.

recordingName

Type: **string**

The name of the recording. Will be an automatically generated name if no name has been supplied.

state

Type: **string**

The state of the streaming playback connection.

Value	Description
pending	The connection is pending streaming playback.
active	The connection is currently streaming playback.
paused	The streaming playback on this connection is currently paused.

playbackPosition

Type: **integer**

Playback position in seconds

mediaDuration

Type: **integer**

The duration in seconds of this playback or recording.

viewer

Type: **array**

A collection of information about the viewer of this streaming playback.

viewer structure

ipAddress

Type: **string**

IPv4 IP address in dotted-quad format.

player

Type: **string**

Player identification

audioCodec

Type: **string**

The codec used on the audio stream. This parameter is only returned when the streaming protocol is either RTSP or MMS.

videoCodec

Type: **string**

The video codec for this streaming connection. This parameter is only returned when the protocol is either RTSP or MMS.

audioTransport

Type: **string**

The transport protocol used for audio on this streaming connection.

Value	Description
udp	The stream is transported on the User Datagram Protocol (UDP)
tcp	The stream is transported on the Transmission Control Protocol (TCP)
http	The stream is transported on the HyperText Transfer Protocol (HTTP)

videoTransport

Type: **string**

The transport protocol used on this streaming video connection.

Value	Description
udp	The stream is transported on the User Datagram Protocol (UDP)
tcp	The stream is transported on the Transmission Control Protocol (TCP)
http	The stream is transported on the HyperText Transfer Protocol (HTTP)

connection.streaming.terminate

Status: **active**

Terminates a streaming playback connection from the device using the `connectionId` and `protocol` to fully identify the streaming connection. Returns fault code 28 (no such connection) if a connection with the supplied `connectionId` does not exist.

Accepts:

connectionId

Type: **integer**

Unique identifier of the connection.

protocol

Type: **string**

The streaming protocol used on this connection.

Value	Description
rtsp	Connection uses the Real Time Streaming Protocol (RTSP)
mms	Connection uses the Microsoft Media Services protocol (MMS)
content	

connection.terminate

Status: **active**

Terminates an active connection on the device using the `connectionId`. Returns fault code 28 (no such connection) if the connection ID does not exist.

Accepts:

`connectionId`

Type: **integer**

Unique identifier of the connection.

device.disk.query

Status: **active**

Returns disk usage.

Returns:

`totalSize`

Type: **integer**

The device's total disk space in kilobytes.

`available`

Type: **integer**

The device's available disk space in kilobytes.

device.health.query

Status: **active**

Returns the current status of the device, such as health monitors and CPU load.

Returns:

`cpuLoad`

Type: **integer**

The CPU load as a percentage of the maximum.

mediaLoad

Type: **integer**

A percentage value representing the proportion of the device's media processing capacity that is currently in use.

audioLoad

Type: **integer**

A percentage value representing the proportion of the device's audio processing capacity that is currently in use.

videoLoad

Type: **integer**

A percentage value representing the proportion of the device's video processing capacity that is currently in use.

temperatureStatus

Type: **string**

Value	Description
ok	The temperature is currently within the normal operating range.
outOfSpec	The temperature is currently outside the normal operating range.
critical	The temperature is too high and the device will shutdown if this condition persists.

temperatureStatusWorst

Type: **string**

The worst temperature status recorded on this device since it booted.

Value	Description
ok	The temperature has been within the normal operating range since the device was booted.
outOfSpec	The temperature has been outside the normal operating range at least once since the device was booted.
critical	At some point since the last boot the temperature was too high. The device will shutdown if this condition persists.

rtcBatteryStatus

Type: **string**

The current status of the RTC battery (Real Time Clock).

Value	Description
ok	The battery is operating within the normal range.
outOfSpec	The battery is operating outside of the normal range, and may require service.

rtcBatteryStatusWorst

Type: **string**

The worst recorded status of the RTC battery.

Value	Description
ok	The battery has been operating inside the normal range since the device was booted.
outOfSpec	The battery has operated outside of the normal range at some time since the device was booted.

voltagesStatus

Type: **string**

Value	Description
ok	The voltage is currently within the normal range
outOfSpec	The voltage is currently outside the normal range

voltagesStatusWorst

Type: **string**

Value	Description
ok	The voltage has been within the normal range since the device last booted.
outOfSpec	The voltage has been outside the normal range at some time since the device last booted

operationalStatus

Type: **string**

Value	Description
active	
shuttingDown	
shutDown	

device.network.query

Status: **active**

Queries the device for its network information. Some of the data listed below will be omitted if the interface is not enabled or configured. The query returns empty strings for addresses that are not configured.

Returns:

portA

Type: **array**

A structure that contains configuration and status information for Ethernet port A on the device.

portB

Type: **array**

A structure that contains configuration and status information for Ethernet port B on the device.

Port arrays structure

enabled

Type: **boolean**

Value	Description
true	The port is enabled.
false	The port is not enabled.

linkStatus

Type: **boolean**

Value	Description
true	The ethernet connection to this port is active.
false	The ethernet connection to this port is not active.

speed

Type: **integer**

Speed of the connection on this Ethernet interface. One of 10, 100 or 1000, in Mbps.

fullDuplex

Type: **boolean**

Value	Description
true	The port can support a full-duplex connection.
false	The port can support a half-duplex connection.

macAddress

Type: **string**

The MAC address of this interface. A 12 character string of hex digits with no separators.

packetsSent

Type: **integer**

The number of packets sent from this Ethernet port.

packetsReceived

Type: **integer**

The number of packets received on this Ethernet port.

multicastPacketsSent

Type: **integer**

Number of multicast packets sent from this Ethernet interface.

multicastPacketsReceived

Type: **integer**

Number of multicast packets received on this Ethernet interface.

bytesSent

Type: **integer**

The number of bytes sent by the device.

bytesReceived

Type: **integer**

The number of bytes received by the device.

queueDrops

Type: **integer**

Number of packets dropped from the queue on this network interface.

collisions

Type: **integer**

Count of the network collisions recorded by the device.

transmitErrorsType: **integer**

The count of transmission errors on this Ethernet interface.

receiveErrorsType: **integer**

The count of receive errors on this interface.

bytesSent64Type: **string**

64 bit versions of the bytesSent statistic, using a string rather than an integer.

bytesReceived64Type: **string**

64 bit versions of the bytesReceived statistic, using a string rather than an integer.

The following parameters are returned only if the interface is enabled and configured.

hostNameType: **string**

The host name of queried device.

dhcpType: **boolean**

Value	Description
true	The device's IP address is allocated by DHCP
false	The device's IP address is manually configured

ipAddressType: **string**

IPv4 IP address in dotted-quad format.

subnetMaskType: **string**

The IPv4 subnet mask in dotted quad format.

defaultGatewayType: **string**

The device's IPv4 default gateway in dotted quad format.

domainNameType: **string**

The domain name (DNS suffix).

nameServerType: **string**The IP address of the name server, in dotted quad format.

The IP address of the name server, in dotted quad format.

nameServerSecondaryType: **string**

The IP address of the secondary name server, in dotted quad format.

device.queryStatus: **active**

Returns high level status information about the device.

Returns:**currentTime**Type: **dateTime.iso8601**

The system's current time (UTC).

restartTimeType: **dateTime.iso8601**

The date and time when the system was last restarted.

serialType: **string**

The serial number of the device.

softwareVersionType: **string**

The version number of the software running on the device.

buildVersionType: **string**

The build version of the software running on the device.

modelType: **string**

The model number.

apiVersion

Type: **string**

The version number of the API implemented by this device.

activatedFeatures

Type: **array**

Each member contains a string named **feature** containing a short description of that feature, for example, **Encryption**.

totalVideoPorts

Type: **integer**

The total number of video ports on the device.

Deprecated in IP VCR API version 2.7.

totalAudioOnlyPorts

Type: **integer**

The total number of additional audio-only ports on the device.

Deprecated in IP VCR API version 2.7.

maxVideoResolution

Type: **string**

Value	Description
cif	The maximum video resolution is 352 x 288
4cif	The maximum video resolution is 704 x 576

totalPlaybackPorts

Type: **integer**

The number of ports this device uses for playback.

totalRecordingPorts

Type: **integer**

The number of ports this device uses for recording.

device.restartlog.query

Status: **active**

Returns the restart log - also known as the system log on the web interface.

Returns:**log**

Type: **array**

Each member of the array contains log information (called system log in the user interface).

log array members**time**

Type: **dateTime.iso8601**

The time when the device restarted.

Value	Description
20110119T13:52:42	yyyymmddThh:mm:ss

reason

Type: **string**

Value	Description
unknown	The software is unaware why the device restarted.
User requested shutdown	The device restarted normally after a user initiated a shutdown.
User requested upgrade	The device restarted itself because a user initiated an upgrade.

folder.enumerate

Status: **active**

This function enumerates all subfolders of a folder.

Accepts:**enumerateID**

Type: **string**

Enumerate calls may return many results so all of them will accept this parameter and may include this parameter in the response.

If the response includes an enumerateID, the application should pass the ID to the subsequent enumerate call to retrieve the next set of responses. If the response does not include an enumerateID, there are no more results in the enumeration.

If the application omits the enumerateID, the target device will start a new enumeration and return the first set of results.

lastRevision

Type: **integer**

This number identifies an earlier set of enumeration data to compare with your current call.

If you supply this parameter using the **currentRevision** value that was returned by a previous enumeration, the current **enumerate** call returns only the differences since that previous call.

If you don't supply this parameter, the device assumes that you want a full enumeration.

listAll

Type: **boolean**

Set this to **true** to do a full enumeration, or **false** to return only a subset (use with **lastRevision** parameter).

Returns:

currentRevision

Type: **integer**

A number that indicates the current revision of this enumeration. You can use this as a **lastRevision** input to a future **enumerate** call to retrieve only the changes between the two enumerations.

enumerateID

Type: **string**

Enumerate calls may return many results so all of them will accept this parameter and may include this parameter in the response.

If the response includes an **enumerateID**, the application should pass the ID to the subsequent **enumerate** call to retrieve the next set of responses. If the response does not include an **enumerateID**, there are no more results in the enumeration.

If the application omits the **enumerateID**, the target device will start a new enumeration and return the first set of results.

folders

Type: **array**

Each member of this array contains information about a specific folder.

folders array

The **folders** array contains a structure for each folder, with each structure containing the following information members:

folderName

Type: **string**

The name of the folder.

parentFolderId

Type: **integer**

The unique identifier for the parent folder of this folder. This is not present if the folder has no parent (it is the top level folder).

externalPath

Type: **string**

The external NFS path to the folder. This value is not present if an external path is not configured.

exportRecordings

Type: **boolean**

This parameter is only present if an external path is configured.

Value	Description
true	Recordings in this folder are exported via NFS
false	Recordings in this folder are not exported via NFS

registerWithGatekeeper

Type: **boolean**

Value	Description
true	This recording is or will be registered with the H.323 gatekeeper
false	This recording is not registered, or it will not be registered, with the H.323 gatekeeper.

public

Type: **boolean**

Value	Description
true	This folder is publicly accessible
false	This folder is not publicly accessible

pin

Type: **string**

The PIN for this item.

autoAttendantId

Type: **string**

The numerical ID used to access the AutoAttendant. **autoAttendantId** must be unique across the IP VCR and may not be the same as other IDs, for example, **recordingId**.

recordingIdType: **integer**

Unique identifier of the recording.

pointToPointIncomingPrefixType: **string**

The folder's incoming point-to-point recording prefix.

pointToPointOutgoingPrefixType: **string**

The folder's outgoing point-to-point recording prefix.

gatekeeper.queryStatus: **active**

Retrieves the gatekeeper settings and current status of the device.

Returns:**gatekeeperUsage**Type: **string**

Value	Description
disabled	The gatekeeper is not used.
enabled	The gatekeeper is used but, if it can't match the call, the call is attempted anyway.
required	The gatekeeper must be used to match the call.

The following parameters are not present if **gatekeeperUsage** is **disabled**.**address**Type: **string**

The address of the item, e.g. endpoint or gateway; may be hostname, IP address or e164 number.

dnsStatusType: **string**

The status of the DNS lookup of the gatekeeper's address.

Value	Description
inProgress	The device is currently trying to resolve the gatekeeper's address

Value	Description
resolved	The gatekeeper's address has been resolved to an IP address
failed	The DNS lookup failed

ip

Type: **string**

the IP address of the gatekeeper (if `dnsStatus` is `resolved`)

activeRegistrations

Type: **integer**

The number of active registrations.

pendingRegistrations

Type: **integer**

The number of registrations in progress

registrationPrefix

Type: **string**

A string of digits that serves as the device's registration prefix.

h323ID

Type: **string**

The H.323 ID used by the device.

mcuServicePrefix

Type: **string**

The service prefix used by the device.

h323IDStatus

Type: **string**

The current status of the ID / service prefix registration process.

Value	Description
idle	
registering	
registered	
deregistering	
pendingReregistration	

Value	Description
waitingRetry	
noID	
idTooLong	

mcuServicePrefixStatus

Type: **string**

The current status of the ID / service prefix registration process.

Value	Description
idle	
registering	
registered	
deregistering	
pendingReregistration	
waitingRetry	
noID	
idTooLong	

gateway.enumerate

Status: **active**

Enumerates configured H.323 gateways on the device.

Accepts:

enumerateID

Type: **string**

Enumerate calls may return many results so all of them will accept this parameter and may include this parameter in the response.

If the response includes an enumerateID, the application should pass the ID to the subsequent enumerate call to retrieve the next set of responses. If the response does not include an enumerateID, there are no more results in the enumeration.

If the application omits the enumerateID, the target device will start a new enumeration and return the first set of results.

Returns:

enumerateID

Type: **string**

Enumerate calls may return many results so all of them will accept this parameter and may include this parameter in the response.

If the response includes an `enumerateID`, the application should pass the ID to the subsequent `enumerate` call to retrieve the next set of responses. If the response does not include an `enumerateID`, there are no more results in the enumeration.

If the application omits the `enumerateID`, the target device will start a new enumeration and return the first set of results.

gateways

Type: **array**

A collection of structures, each of which describes a gateway.

gateways array

name

Type: **string**

The name of the enumerated item, e.g. gateway or endpoint.

address

Type: **string**

The address of the item, e.g. endpoint or gateway; may be hostname, IP address or e164 number.

conferencingParameters

Type: **struct**

A structure containing the conferencing parameters of the enumerated item, e.g. gateway or endpoint.

conferencingParameters array

useDefaultMotionSharpness

Type: **boolean**

Value	Description
true	This endpoint will use box-wide default motion sharpness settings.
false	This endpoint will not use box-wide default motion sharpness settings.

minFrameRateMotionSharpness

Type: **integer**

Specifies the minimum frame rate for this endpoint. This parameter is only present if `useDefaultMotionSharpness` is `false`.

maxMediaTxBitRateType: **integer**

The maximum media transmission speed from this device, in kbps.

maxMediaRxBitRateType: **integer**

The maximum media reception speed of this device, in kbps.

recording.calloutStatus: **active**

This command replicates the call out and record functionality from the web interface.

May return an operation failed fault, with a reason given in the fault string, if the operation fails. Otherwise, the success response includes the integer **recordingId** to uniquely identify the new recording. The application can use the value in later calls such as **recording.configure** or **recording.stop**.

Accepts:**address**Type: **string**

The address of the item, e.g. endpoint or gateway; may be hostname, IP address or e164 number.

folderIdType: **integer**

A numeric reference to the target folder. If no value is supplied, the top level folder is used.

gatewayAddressType: **string**

The address of an H.323 gateway, if required. Only used if protocol is **h323**. This corresponds to the **address** parameter of the gateway as returned by **gateway.enumerate**.

participantProtocolType: **string**

Value	Description
h323	The participant joins using the H.323 protocol (default value)
sip	The participant joins using the SIP protocol

recordingNameType: **string**

The name of the recording. Will be an automatically generated name if no name has been supplied.

useSIPRegistrar

Type: **boolean**

Not valid unless the protocol is SIP.

Value	Description
true	The endpoint uses the SIP registrar
false	The endpoint does not use the SIP registrar (default)

Returns:

recordingId

Type: **integer**

Unique identifier of the recording.

recording.configure

Status: **active**

Sets the parameters of an existing recording.

Accepts:

recordingId

Type: **integer**

Unique identifier of the recording.

recordingName

Type: **string**

The name of the recording. Will be an automatically generated name if no name has been supplied.

numericId

Type: **string**

The numeric ID used for this recording.

pin

Type: **string**

The PIN for this item.

registerWithSIPRegistrar

Type: **boolean**

Value	Description
true	This recording is or will be registered with the SIP registrar
false	This recording is not registered with the SIP registrar

registerWithGatekeeper

Type: **boolean**

Value	Description
true	This recording is or will be registered with the H.323 gatekeeper
false	This recording is not registered, or it will not be registered, with the H.323 gatekeeper.

playbackEnabled

Type: **boolean**

Value	Description
true	This recording has playback enabled
false	This recording does not have playback enabled

recording.delete

Status: **active**

Deletes the specified recording. Returns a fault (no such recording) if the supplied **recordingId** does not match any of the recordings.

Accepts:

recordingId

Type: **integer**

Unique identifier of the recording.

recording.enumerate

Status: **active**

Enumerates the recordings on the IP VCR. You can specify an **enumerateID** to retrieve a specific subset of recordings. You can also filter the response by supplying criteria with the call.

The IP VCR responds with an array of recordings; for each recording it returns a **struct** containing the information it has about the recording.

Accepts:**enumerateID**

Type: **string**

Enumerate calls may return many results so all of them will accept this parameter and may include this parameter in the response.

If the response includes an `enumerateID`, the application should pass the ID to the subsequent `enumerate` call to retrieve the next set of responses. If the response does not include an `enumerateID`, there are no more results in the enumeration.

If the application omits the `enumerateID`, the target device will start a new enumeration and return the first set of results.

enumerateFilter

Type: **string**

A filter expression. The enumeration results depend on the supplied expression.

lastRevision

Type: **integer**

This number identifies an earlier set of enumeration data to compare with your current call.

If you supply this parameter using the `currentRevision` value that was returned by a previous enumeration, the current `enumerate` call returns only the differences since that previous call.

If you don't supply this parameter, the device assumes that you want a full enumeration.

listAll

Type: **boolean**

Set this to `true` to do a full enumeration, or `false` to return only a subset (use with `lastRevision` parameter).

Filters on:**recordingId**

Type: **integer**

Unique identifier of the recording.

internal

Type: **boolean**

Value	Description
true	This recording is stored internally (on the device)
false	This recording is not stored internally

external

Type: **boolean**

Value	Description
true	Include externally stored recordings in the filtered results.
false	Exclude externally stored recordings from the filtered results.

inProgress

Type: **boolean**

Value	Description
true	The recording is active.
false	The recording is not currently taking place.

connectionId

Type: **integer**

Unique identifier of the connection.

Returns:

currentRevision

Type: **integer**

A number that indicates the current revision of this enumeration. You can use this as a **lastRevision** input to a future **enumerate** call to retrieve only the changes between the two enumerations.

recordings

Type: **array**

The **recordings** array contains a structure for each recording. The structure members contain information about the recording. The **connectionId** is present if the recording is currently active.

Recording structure

recordingId

Type: **integer**

Unique identifier of the recording.

connectionId

Type: **integer**

Unique identifier of the connection.

recordingNameType: **string**

The name of the recording. Will be an automatically generated name if no name has been supplied.

folderIdType: **integer**

A numeric reference to the target folder. If no value is supplied, the top level folder is used.

externalType: **boolean**

Value	Description
true	Include externally stored recordings in the filtered results.
false	Exclude externally stored recordings from the filtered results.

numericIdType: **string**

The numeric ID used for this recording.

pinType: **string**

The PIN for this item.

statusType: **string**

The current status of the recording.

Value	Description
idle	The recording is not in any of the other states.
initialising	The IP VCR will soon begin recording to this file.
invalid	The recording file is invalid.
uploading	A recording is being uploaded to the IP VCR.
deleting	A recording is being deleted from the IP VCR.
recording	The IP VCR is currently recording to this file.

playbackEnabledType: **boolean**

Value	Description
true	This recording has playback enabled
false	This recording does not have playback enabled

registerWithSIPRegistrar

Type: **boolean**

Value	Description
true	This recording is or will be registered with the SIP registrar
false	This recording is not registered with the SIP registrar

registerWithGatekeeper

Type: **boolean**

Value	Description
true	This recording is or will be registered with the H.323 gatekeeper
false	This recording is not registered, or it will not be registered, with the H.323 gatekeeper.

recording.play

Status: **active**

Play back a recording to an endpoint, returns `connectionId` when successful.

Accepts:

recordingId

Type: **integer**

Unique identifier of the recording.

address

Type: **string**

The address of the item, e.g. endpoint or gateway; may be hostname, IP address or e164 number.

participantProtocol

Type: **string**

Value	Description
h323	The participant joins using the H.323 protocol (default value)

Value	Description
sip	The participant joins using the SIP protocol

gatewayAddress

Type: **string**

The address of an H.323 gateway, if required. Only used if protocol is **h323**. This corresponds to the **address** parameter of the gateway as returned by **gateway.enumerate**.

useSIPRegistrar

Type: **boolean**

Not valid unless the protocol is SIP.

Value	Description
true	The endpoint uses the SIP registrar
false	The endpoint does not use the SIP registrar (default)

Returns:

connectionId

Type: **integer**

Unique identifier of the connection.

recording.stop

Status: **active**

Stops the specified recording. Returns a fault (no such recording) if the supplied **recordingId** does not match any of the recordings.

Accepts:

recordingId

Type: **integer**

Unique identifier of the recording.

sip.query

Status: **active**

Retrieves information about SIP configuration on the device.

Returns:**configuredRegistrar**Type: **string**

The currently configured SIP registrar address. This corresponds to the SIP registrar address on the **Settings** web page, and will be an empty string value if there is no currently configured SIP registrar.

configuredProxyType: **string**

The currently configured SIP proxy address. This corresponds to the **SIP proxy address** on the **Settings** web page, and will be an empty string value if there is no currently configured SIP proxy.

registrarContactURIType: **string**

The URI provided to the SIP registrar to register this device.

registrarContactDomainType: **string**

The domain portion of the URI used to register.

Feedback receivers

The interface between the calling application and the Cisco TelePresence device is stateless, so the API allows you to register your application as a feedback receiver. This means that the application doesn't have to constantly poll the device if it wants to monitor activity.

The device publishes events when they occur. If the device knows that your application is listening for these events, it will send XML-RPC messages to your application's interface when the events occur.

Use [feedbackReceiver.configure](#) to register a receiver to listen for one or more [feedback events](#).

Use [feedbackReceiver.query](#) to return a list of receivers that are configured on the device.

After registering as a feedback receiver, the application will receive [feedback messages](#) on the specified interface.

feedbackReceiver.configure

Status: **active**

This call configures the device to send feedback about the specified **events** to the specified **receiverURI**. See the [list of feedback events](#) when you define the **events** array.

Accepts:

receiverURI

Type: **string**

Fully-qualified URI that identifies the listening application's XML-RPC interface (protocol, address, and port), for example, `http://tms1:8080/RPC2`. If this parameter is absent or empty, then the receiver will not receive notifications. You can use `http` or `https` and, if no port number is specified, the device will use the protocol defaults (80 and 443 respectively).

sourceIdentifier

Type: **string**

The device will use this optional parameter in feedback messages to this receiver (event notifications and `feedbackReceiver.query` responses). If this parameter is absent, the device uses the MAC address of its Ethernet port A interface.

receiverIndex

Type: **integer**

Sets the position of this feedback receiver in the device's table of feedback receivers.

Value	Description
-1	The feedback receiver will use any available position.
1	The first position is assumed if you don't supply this parameter (will overwrite existing entry in position 1)
20	The 20th (maximum allowed) position

events

Type: **struct**

An associative array mapping strings to booleans, where the string is the name of the feedback event and the boolean indicates if the feedback receiver wants to receive a notification of that event.

If this parameter is absent, then the receiver will be configured to receive the default notification messages (all notifications).

feedbackReceiver.query

Status: **active**

This call asks the device for a list of all the feedback receivers that have previously been configured. It does not accept parameters other than the authentication strings.

Returns:

receivers

Type: **array**

An array of feedback receivers, with members corresponding to the entries in the receivers table on the device's web interface.

Receiver members

Each receiver in the response contains the following parameters:

receiverURI

Type: **string**

Fully-qualified URI that identifies the listening application's XML-RPC interface (protocol, address, and port), for example, `http://tms1:8080/RPC2`. If this parameter is absent or empty, then the receiver will not receive notifications. You can use `http` or `https` and, if no port number is specified, the device will use the protocol defaults (80 and 443 respectively).

sourceIdentifier

Type: **string**

The device will use this optional parameter in feedback messages to this receiver (event notifications and `feedbackReceiver.query` responses). If this parameter is absent, the device uses the MAC address of its Ethernet port A interface.

index

Type: **integer**

The position in the device's table of feedback receivers that this receiver uses. A number between 1 and 20 (inclusive).

Feedback messages

The feedback messages follow the format used by the device for XML-RPC responses.

The messages contain two parameters:

- **sourceIdentifier** is a string that identifies the device, which may have been set by **feedbackReceiver.configure** or otherwise will be the device's MAC address.
- **events** is an array of strings that contain the names of the feedback events that have occurred.

Example feedback message

```
<params>
  <param>
    <value>
      <struct>
        <member>
          <name>sourceIdentifier</name>
          <value><string>000D7C000C66</string></value>
        </member>
        <member>
          <name>events</name>
          <value>
            <array>
              <data>
                <value><string>restart</string></value>
              </data>
            </array>
          </value>
        </member>
      </struct>
    </value>
  </param>
</params>
```

Feedback events

The following table lists the feedback events that this device can publish.

Event	Description
autoAttendantStarted	An auto attendant has started
autoAttendantFinished	An auto attendant has finished
playbackStarted	Playback has started
playbackFinished	Playback has finished
playbackStreamingStarted	Streaming playback has started
playbackStreamingFinished	Streaming playback has finished
recordingStarted	Recording has started
recordingFinished	Recording has finished

Event	Description
recordingUploadStarted	Recording upload has started
recordingUploadFinished	Recording upload has finished
recordingStreamingStarted	Live streaming viewer has started
recordingStreamingFinished	Live streaming viewer has finished
recordingAdded	Recording has been added to an external folder
recordingRemoved	Recording has been removed from an external folder
recordingDeleted	Recording has been deleted
recordingMoved	Recording has been moved to a different folder
recordingExported	Recording has been exported to an external folder
recordingNameChanged	Recording has been renamed
recordingConfigurationChanged	Recording configuration has changed
folderCreated	Folder has been created
folderDeleted	Folder has been deleted
folderMoved	Folder has been moved to a different folder
folderNameChanged	Folder has been renamed
folderConfigurationChanged	Folder configuration has changed

Related information

Fault codes

Many of the software applications that run on Cisco TelePresence hardware will return a fault code when they encounter a problem while processing an XML-RPC request.

The following table describes all the fault codes used and their most common interpretations. Not all codes are used by all products.

Fault Code	Description
1	Method not supported. This method is not supported on this device.
2	Duplicate conference name. A conference name was specified, but is already in use.
3	Duplicate participant name. A participant name was specified, but is already in use.
4	No such conference or auto attendant. The conference or auto attendant identification given does not match any conference or auto attendant.
5	No such participant. The participant identification given does not match any participants.
6	Too many conferences. The device has reached the limit of the number of conferences that can be configured.
7	Too many participants. There are already too many participants configured and no more can be created.
8	No conference name or auto attendant id supplied. A conference name or auto attendant identifier was required, but was not present.
9	No participant name supplied. A participant name is required but was not present.
10	No participant address supplied. A participant address is required but was not present.
11	Invalid start time specified. A conference start time is not valid.
12	Invalid end time specified. A conference end time is not valid.
13	Invalid PIN specified. A PIN specified is not a valid series of digits.
14	Unauthorised. The requested operation is not permitted on this device.
15	Insufficient privileges. The specified user id and password combination is not valid for the attempted operation.
16	Invalid enumerateID value. An enumerate ID passed to an enumerate method invocation was invalid. Only values returned by the device should be used in enumerate methods.
17	Port reservation failure. This is in the case that reservedAudioPorts or reservedVideoPorts value is set too high, and the device cannot support this.
18	Duplicate numeric ID. A numeric ID was given, but this ID is already in use.
19	Unsupported protocol. A protocol was used which does not correspond to any valid protocol for this method. In particular, this is used for participant identification where an invalid protocol is specified.
20	Unsupported participant type. A participant type was used which does not correspond to any participant type known to the device.

21	No such folder. A folder identifier was present, but does not refer to a valid folder.
22	No such recording. A recording identifier was present, but does not refer to a valid recording.
23	No changes requested. This is given when a method for changing something correctly identifies an object, but no changes to that object are specified.
24	No such port. This is returned when an ISDN port is given as a parameter which does not exist on an ISDN gateway.
25	New port limit lower than currently active
26	Floor control not enabled for this conference
27	No such template. The specified template wasn't found.
28	No such connection. The specified connection was not found.
30	Unsupported bit rate. A call tried to set a bit rate that the device does not support.
31	Template name in use. This occurs when trying to create or rename a template to have the same name as an existing template.
32	Too many templates. This occurs when trying to create a new template after the limit of 100 templates has been reached.
33	Parameter value is not inside the expected range. A call supplied a value that is outside of the allowed range for this parameter.
34	Internal error
35	String is too long. The call supplied a string parameter that was longer than allowed. Strings are usually limited to 32 characters but may be 63 in some cases.
101	Missing parameter. This is given when a required parameter is absent. The parameter in question is given in the fault string in the format "missing parameter - parameter_name".
102	Invalid parameter. This is given when a parameter was successfully parsed, is of the correct type, but falls outside the valid values; for example an integer is too high or a string value for a protocol contains an invalid protocol. The parameter in question is given in the fault string in the format "invalid parameter - parameter_name".
103	Malformed parameter. This is given when a parameter of the correct name is present, but cannot be read for some reason; for example the parameter is supposed to be an integer, but is given as a string. The parameter in question is given in the fault string in the format "malformed parameter - parameter_name".
201	Operation failed. This is a generic fault for when an operation does not succeed as required.

HTTP keep-alives

Note: This feature is available from API version 2.4 onwards.

Your application can use use HTTP keep-alives to reduce the amount of TCP traffic that results from constantly polling the device. Any client which supports HTTP keep-alives may include the following line in the HTTP header of an API request:

Connection: Keep-Alive

This indicates to the device that the client supports HTTP keep-alives. The device may then choose to maintain the TCP connection after it has responded. If the device will close the connection it returns the following HTTP header in its response:

Connection: close

If this line is not in the HTTP header of the response, the client may use the same connection for a subsequent request.

The device will not keep a connection alive if:

- the current connection has already serviced the allowed number of requests
- the current connection has already been open for the allowed amount of time
- the number of open connections exceeds the allowed number if this connection is maintained

These restrictions are in place to limit the resources associated with open connections. If a connection is terminated for either of the first two reasons, the client will probably find that the connection is maintained after the next request.

Note: The client should never assume a connection will be maintained. Also, the device will close an open connection if the client does not make any further requests within a minute. There is little benefit to keeping unused connections open for such long periods.

system.xml file

You can derive some information about the device from its **system.xml** file. You can download this file via HTTP from the device's root.

Example system.xml

```
<?xml version="1.0"?>
  <system>
    <manufacturer>Codian</manufacturer>
    <model>MSE 8220</model>
    <serial>SM0100A9</serial>
    <softwareVersion>3.0(1.11)</softwareVersion>
    <buildVersion>7.3(1.11)</buildVersion>
    <hostName></hostName>
    <totalRecordingPorts>10</totalRecordingPorts>
    <totalPlaybackPorts>20</totalPlaybackPorts>
    <maxVideoResolution>max</maxVideoResolution>
    <uptimeSeconds>480</uptimeSeconds>
  </system>
```

Checking for updates and getting help

We recommend registering your product at <http://www.tandberg.com/services/video-conferencing-product-registration.jsp> in order to receive notifications about the latest software and security updates. New feature and maintenance releases are published regularly, and we recommend that your software is always kept up to date.

If you experience any problems when configuring or using the product, consult the documentation at <http://www.tandberg.com/support/video-conferencing-documentation.jsp> for an explanation of how its individual features and settings work. You can also check the support site at <http://www.tandberg.com/support/> to make sure you are running the latest software version.

You or your reseller can also get help from our support team by raising a case at <http://www.tandberg.com/support/>. Make sure you have the following information ready:

- The software build number which can be found in the product user interface (if applicable).
- Your contact email address or telephone number.
- The serial number of the hardware unit (if applicable).

References

1. XML-RPC specification (Dave Winer, June 1999); <http://www.xmlrpc.com/spec>, accessed 24/01/2011.
2. HTTP/1.1 specification (RFC 2616, Fielding et al., June 1999); <http://www.ietf.org/rfc/rfc2616.txt>, accessed 24/01/2011.

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Cisco and the Cisco Logo are trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and other countries. A listing of Cisco's trademarks can be found at www.cisco.com/go/trademarks. Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1005R)

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

© 2011 Cisco Systems, Inc. All rights reserved.