Cisco TelePresence VCR Remote Management API

Reference Guide

D14661.03

December 2010

Contents

API history

The latest version of the Cisco TelePresence VCR Remote Management API is version 2.5. The following Cisco TelePresence VCR products support this API version when running software version 2.3, and later:

- IP VCR 2200 Series
- VCR MSE 8220

The following table shows which version of Cisco TelePresence products support which version.

API Version	IP VCR 2210, IP VCR 2220, IP VCR 2240, VCR MSE 8220
2.4	2.2
2.5	2.3, and later

Introduction

This reference guide contains the specification of the Cisco TelePresence VCR Remote Management API, by which it is possible to control the following Cisco TelePresence products:

- IP VCR 2210, 2220 and 2240
- VCR MSE 8220

This is accomplished via messages sent using the XML-RPC protocol. XML-RPC is a simple protocol for remote procedure calling using HTTP as the transport and XML as the encoding. It is designed to be as simple as possible, whilst allowing complex data structures to be transmitted, processed and returned. XML-RPC has no platform or software dependence and was chosen over SOAP because of its simplicity.

The interface is stateless. Currently, there is no mechanism for the Cisco TelePresence VCR (VCR) to call back the controlling application and therefore the controlling application must poll the VCR for status, as required. A future enhancement *may* provide a mechanism for signaling device status changes to the controlling application.

In this implementation of XML-RPC all parameters and return values are part of a <struct> and are explicitly named. For example, the device.query call returns the current time value as a structure member named 'currentTime' rather than as a single value of type <dateTime.iso8601>.

Note: Unless otherwise stated, assume string length is a maximum of 32 characters.

For further details of XML-RPC refer to the specification [1].

HTTP and HTTPS

VCRs expect to receive HTTP communication over TCP/IP connections to port 80. The HTTP messages should be "POST"s to the URL "/RPC2".

HTTPS (a secure, encrypted version of HTTP) is supported on all VCR products from software version 2.3 and later.

By default, HTTPS is provided on TCP port 443, although VCRs can be configured to receive HTTP and HTTPS connections on non-standard TCP port numbers if required.

The VCR implements HTTP/1.1 as defined by RFC 2616 [2].

Consider API overhead when writing applications

Every API command that your application sends incurs a processing overhead within the VCR's own application. The exact amount of overhead varies widely with the command type and the parameters sent. It is important to bear this in mind when designing your application's architecture and software. If the device receives a high number of API commands every second, its overall performance could be seriously impaired – in the same way that it would if be several users accessed it from the web interface simultaneously.

For this reason, the best architecture is a single server running the API application and sending commands to the VCR. If multiple users need to use the application simultaneously, provide a web interface on that server or write a client that communicates with the server. The server would then manage the clients' requests and send API commands directly to the VCR. Implement some form of control in the API application on your server to prevent the VCR being overloaded with API commands. This provides much more control than having the clients send API commands directly and will prevent the VCR's performance being impaired by unmanageable numbers of API requests.

Protocol overview

Authentication

In order to manage the VCR, the controlling application must authenticate itself as a user with relevant privileges. Accordingly, each message contains a user name and password; see the section *Common message elements* below for details of the format. Note that authentication information is sent using plain text and should only be sent over a trusted network.

Note: All calls require administrator privileges.

Message flow

An application can send command messages to the VCR. For each command sent (provided the message is correctly formatted according to the XML-RPC spec), the VCR responds with a message indicating success or failure. The response message may also contain any data that was requested.

Command messages are sent in XML format. For example, the following message deletes a recording on a VCR.

```
POST /RPC2 HTTP/1.0
Host: dt1b5.test.taa
User-Agent: xmlrpclib.py/1.0.1 (by www.pythonware.com)
Content-Type: text/xml
Content-Length: 429

<?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>
```

If the command was successful, the VCR sends a success response.

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/xml
Content-Encoding: utf8
Content-Length: 240
```

```
<?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>
```

If the command fails, for example, trying to delete a recording that does not exist, the VCR sends a fault response, for example:

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/xml
Content-Encoding: utf8
Content-Length: 404
<?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>
```

The complete list of command messages, their required and optional parameters, and the expected responses are detailed in the sections below. The possible fault codes are listed later in this document.

Unicode support

Parameters in this version of the API can be in ASCII text or unicode (UTF-8). In order to distinguish between these encodings, any of several methods can be used. If no method is present, ASCII is assumed.

HTTP headers

There are two different ways of specifying unicode in the HTTP headers; either using "Accept-Encoding: utf-8", or modifying the Content-Type header to read "Content-Type: text/xml; charset=utf-8".

XML header

The **<?xml>** tag is required at the top of each XML file. This API will accept an additional encoding parameter with value UTF-8 for this tag, i.e. <?xml version="1.0" encoding="UTF-8"?>.

Common message elements

Authentication

All messages must contain a user name and password as follows:

Parameter	Type	Comments
authenticationUser	String	Name of a user with sufficient privilege for the operation being performed. The name is case sensitive.
authenticationPassword	String	The corresponding user's password. This parameter is ignored if the user has no password set. Note that this differs from the web interface where a blank password must be blank.

Note: All calls require administrator privileges.

Enumerate functions

Due to the potential for a very large number of responses, all enumerate functions return an enumerateID response. This contains a value which should be passed to subsequent calls of the same enumerate function in order to retrieve the remainder of the values.

The use of this parameter is as follows:

1. The client computer sends an enumerate call with any necessary parameters (e.g. operationScope) and no enumerateID parameter.
2. The VCR returns with an array containing the requested data, and possibly a new enumerateID.
3. If there is an enumerateID, the client will call the enumerate method again, with any parameters that are required or desired, and an enumerateID parameter containing the ID returned by the VCR from the previous call. This should be repeated while the VCR continues to provide new enumerateID values in responses.
4. After all data is returned, the VCR will reply with all remaining results, but no enumerateID.

This method should only be called using enumerateID values as provided by the VCR.

Filtering

Enumerate functions contain an optional enumerateFilter parameter, which can be used to restrict the responses to the enumerate call. The valid expressions depend on the function to which they are applied, but the syntax is the same for all enumerate functions: the section in this document for each function provides a list of valid filters for that function.

To use the filters, the expression is evaluated, with any function or expression symbols evaluated for the given entity being enumerated (e.g. if enumerating recordings, the inProgress expression will evaluate to true if the recording is in the process of being made, and false otherwise). If the result of evaluating the filter is true, the entity is returned to the client. If the expression evaluates to false, the entity in question is not returned to the client and the next entity (if any) is considered. As an example, if the expression (inProgress && internal) is used when enumerating recordings, the returned recordings will be only those which are both inProgress and internal.

Filters can consist of atomic expressions, joined together with operators, and brackets in the traditional manner. Whitespace is ignored. Functions are valid, and any parameters are in a comma separated list in brackets following the function name, for example *function(expression$_1$,expression2)*.

From a boolean perspective, the integer 0 is false, and all other numbers are true.

Integer values can be expressed using any string of valid digits, optionally prefixed by 0x for hexadecimal, 0t for decimal and 0z for binary. If no prefix is specified, decimal is assumed.

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

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

There are also the following unary operators, all of which bind tighter than any binary operator.

Operator	Description
-	Unary minus
+	Unary plus
!	Logical negation
~	Bitwise negation

An example filter would be !(*expression*$_1$ && *expression*$_2$)

API reference

This section is a reference to each of the VCR API calls. For each API call, the following information is provided where applicable:

- Description of the call's function
- Parameters
- List of responses
- Response data types
- Structure formats
- Additional information that will assist in using the API call
- Parameters deprecated from previous versions

The table below lists all currently supported VCR API calls. Click on the call name to go to the page containing a complete description of the call.

addressBookEntry.enumerate

Enumerates configured endpoints on a VCR.

Parameter	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified above. If required, enter the value returned in enumerateID by the most recent addressBookEntry.enumerate response. If it is omitted, a new enumeration is started.

This method returns:

Response	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified above. The value that should be used in the next addressBookEntry.enumerate call to get the next set of data. If this is omitted, no further data is available from the VCR.
addressBookEntries	Array	See below for details.

The array “addressBookEntries” contains structs with the following fields:

Field	Type	Comments
name	String	The configuration’s name.
address	String	The endpoint’s E.164 directory number, hostname or IP address.
protocol	String	One of **`h323`** or **`sip`**.
gatewayName	String	Present for H.323 endpoints which are configured to use a gateway. This name corresponds to the name of a gateway returned via gateway.enumerate.
gatewayAddress	String	Present for H.323 endpoints which are configured to use a gateway. This is the address of the gateway that this endpoint is configured to use.
useSIPRegistrar	Boolean	Whether this endpoint is configured to use a sip registrar when being called.
callInParams	Array	See below for details.
conferencingParameters	Array	See below for details.

The structure callInParams contains the following fields. This is used to match incoming connections to endpoint configurations. For a positive match a connection must match fields which have values. Blank fields are not considered in the comparison.

Field	Type	Comments
name	String	Endpoint name.
address	String	IP address.
e164	String	E.164 number.

The structure conferencingParameters contains the following fields:

Field	Type	Comments
useDefaultMotionSharpness	Boolean	If true, this endpoint will use box-wide default motion sharpness settings.
minFrameRateMotionSharpness	Integer	Only present if useDefaultMotionSharpness is false. Specifies the minimum frame rate for this endpoint.
useDefaultVideoTransmitResolutions	Boolean	If true, this endpoint will use box-wide default video transmit resolutions.
videoTransmitResolutions	String	One of: **`allowAll`**, **`4to3Only`**, **`4to3WidescreenOverride`** or **`16to9Only`**.
maxMediaTxBitRate	Integer	Max media transmit bit rate.
maxMediaRxBitRate	Integer	Max media receive bit rate.
defaultLayout	String	Refer to appendix A for a list of layouts.
layoutControlDefault	Boolean	If true, this endpoint will use box-wide layout control settings.
layoutControlEnabled	Boolean	Only present if layoutControlDefault is false. Indicates whether the endpoint's participant will have control over their layout.
h239ContributionDefault	Boolean	If true, this endpoint will use box-wide H.239 contribution settings.
h239ContributionEnabled	Boolean	Only present if h239ContributionDefault is false. Specifies whether the endpoint will be able contribute H.239.
initialAudioMuted	Boolean	Whether this endpoint would initially have their audio muted.
initialVideoMuted	Boolean	Whether this endpoint would initially have their video muted.
autoDisconnect	Boolean	When an endpoint disconnects from a conference and only endpoints which have autoDisconnect set to true remain, all those endpoints are disconnected.

Field	Type	Comments
borderWidth	Integer	0 (no border), or 1, 2, or 3 for +1/+2/+3.

device.health.query

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

Response	Type	Comments
cpuLoad	Integer	The CPU load, as a percentage.
mediaLoad	Integer	Loads for the media processors (total, and split between audio and video) as percentage values.
audioLoad	Integer	
videoLoad	Integer	
temperatureStatus	String	One of ok, outOfSpec or critical.
temperatureStatusWorst	String	
temperatureStatus	String	
temperatureStatusWorst	String	
rtcBatteryStatus	String	
rtcBatteryStatusWorst	String	
voltagesStatus	String	
voltagesStatusWorst	String	
operationalStatus	String	One of active, shuttingDown or shutDown.

device.network.query

This call takes no parameters. The response returns the following:

Parameter	Type	Comments
portA	Array (see below)	Contains the configuration and status for port A.
portB	Array (see below)	Contains the configuration and status for port B.

The format for the two structures above is:

Field	Type	Comments
enabled	Boolean	true if the port is enabled, otherwise false.
linkStatus	Boolean	true if the link is up, false if the link is down.
speed	Integer	One of 10, 100 or 1000, in Mbps.
fullDuplex	Boolean	true if full duplex enabled, false if half.
macAddress	String	A 12 character string, no separators.
packetsSent	Integer	Stats from the web interface. It is worth noting that all these values are 32 bit signed integers, and thus may wrap.
packetsReceived	Integer	
multicastPacketsSent	Integer	
multicastPacketsReceived	Integer	
bytesSent	Integer	
bytesReceived	Integer	
queueDrops	Integer	
collisions	Integer	
transmitErrors	Integer	
receiveErrors	Integer	

Field	Type	Comments
bytesSent64	String	64 bit versions of the above stats, using a string rather than an integer.
bytesReceived64	String	
Optional parameters		
hostName	String	The host name of this port.
dhcp	Boolean	true if configured by DHCP, otherwise **`false`**.
ipAddress	String	a.b.c.d format.
subnetMask	String	a.b.c.d format.
defaultGateway	String	a.b.c.d format.
domainName	String	The domain name of this port.
nameServer	String	a.b.c.d format.
nameServerSecondary	String	a.b.c.d format.

All fields above marked optional will only be returned if the interface has been enabled and has been configured.

device.query

There are no parameters passed with this method call. The method response returns the following:

Parameter	Type	Comments
currentTime	dateTime.iso8601	The system's current time (UTC).
restartTime	dateTime.iso8601	The date and time at which the system was last restarted.
serial	String	The serial number of the VCR.
softwareVersion	String	The software version of the running software.
buildVersion	String	The build version of the running software.
model	String	The model of this VCR.
apiVersion	String	The version number of the API implemented by this VCR.
activatedFeatures	Array	Currently, only contains a string "feature" with a short description of the feature.
totalVideoPorts	Integer	The total number of video ports on the VCR.
totalAudioOnlyPorts	Integer	The total number of additional audio-only ports on the VCR.
maxVideoResolution	String	One of **`cif`** or **`4cif`**.

device.restartlog.query

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

Response	Type	Comments
log	Array	Contains the restart log in structures as described below.

The "Log" array consists of structures which contain the following fields.

Field	Type	Comments
time	dateTime.iso8601	The time of the last reboot.
reason	String	The reason for the reboot (one of **`unknown`**, **`User requested shutdown`** or **`User requested upgrade`**).

folder.enumerate

This function enumerates all subfolders of a folder.

Parameter	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified above. If required, enter the value returned in enumerateID by the most recent folder.enumerate response. If it is omitted, a new enumeration is started.

This returns the following structure:

Response	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified above.
folders	Array	See below for details.

The "folders" array contains the following structures:

Field	Type	Comments
folderName	String	The name of the folder.
folderId	Integer	A unique identifier for this folder.
parentFolderId (optional)	Integer	The unique identifier for the parent folder of this folder. This is not present if the folder has no parent (i.e. is the top level folder).
externalPath (optional)	String	The external NFS path. Only present if there is an external path configured.
exportRecordings (optional)	Boolean	Set to true if recordings in this folder are exported via NFS. Only present if there is an external path configured.
registerWithGatekeeper (optional)	Boolean	Set to true if the recordings exported externally in this folder are to be registered with the gatekeeper. Only present if there is an external path configured.
public	Boolean	Set to true if this folder is publicly accessible.
pin	String	Contains the PIN of this folder.
autoAttendantId	String	The numerical ids used to access these functions. Note that these are not the same as other instances of similar names, e.g. recordingId. These must be unique across the VCR.
recordingId	String	
recordingConsoleId	String	

Field	Type	Comments
pointToPointIncomingPrefix	String	The folder's configured point-to-point recording prefixes.
pointToPointOutgoingPrefix	String	

gatekeeper.query

Retrieves the gatekeeper settings and current status for a VCR. Takes no parameters.

Response	Type	Comments
gatekeeperUsage	String	One of **`disabled`**, **`enabled`** or **`required`**.
The following parameters are all optional and will only be present if gatekeeperUsage is not **`disabled`**.		
address	String	The address of the gatekeeper.
dnsStatus	String	The status of the DNS resolution: one of **`inProgress`**, **`resolved`** or **`failed`**.
ip	String	If the dnsStatus is **`resolved`**, contains the IP address of the gatekeeper.
activeRegistrations	Integer	The number of active registration.
pendingRegistrations	Integer	The number of registrations in progress.
registrationPrefix	String	The registration prefix used by the VCR
h323ID	String	The H.323 ID used by the VCR.
mcuServicePrefix	String	The service prefix used by the VCR.
h323IDStatus	String	The current status of the ID/service prefix registration process. One of **`idle`**, **`registering`**, **`registered`**, **`deregistering`**, **`pendingReregistration`**, **`waitingRetry`**, **`noID`** or **`idTooLong`**.
mcuServicePrefixStatus	String	

gateway.enumerate

Enumerates configured H.323 gateways on a VCR.

Parameter	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified above. If required, enter the value returned in enumerateID by the most recent gateway.enumerate response. If it is omitted, a new enumeration is started.

This method returns:

Response	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified above. The value which should be used in the next gateway.enumerate call to get the next set of data. If this is omitted, no further data is available from the VCR.
gateways	Array	See below for details.

The array “gateways” contains structs with the following fields:

Field	Type	Comments
name	String	The name of the configured gateway.
address	String	The gateway’s E.164 directory number, hostname or IP address.
conferencingParameters	Array	See below for details.

The structure conferencingParameters contains the following fields:

Field	Type	Comments
useDefaultMotionSharpness	Boolean	If true, this endpoint will use box-wide default motion sharpness settings.
minFrameRateMotionSharpness	Integer	Only present if useDefaultMotionSharpness is false. Specifies the minimum frame rate for this endpoint.
maxMediaTxBitRate	Integer	Max media transmit bit rate.
maxMediaRxBitRate	Integer	Max media receive bit rate.

recording.callout

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

Parameter	Type	Comments
recordingName (optional)	String	The name to be used for the recording. If no name is specified, a default name is used.
folderId (optional)	Integer	The folder in which this recording is to be placed. If not specified, the top level folder will be used.
Address	String	The hostname, IP address or e164 number to connect to.
participantProtocol (optional)	String	Either **`h323`** or **`sip`**. The protocol to be used for this connection. This defaults to **`h323`**.
gatewayAddress (optional)	String	The address of an H.323 gateway, if required. Only used if protocol is **`h323`**.
useSIPRegistrar (optional)	Boolean	Only valid if protocol is **`sip`**. Defaults to false.

This may give an operation failed fault, with a reason given in the fault string, if the operation fails. In VCR release 2.3 and later, if this call succeeds, then the success response includes the integer "recordingId" value used to uniquely identify the new recording. This value can then be used in later recording.configure or recording.stop calls, for instance.

recording.configure

Configures a pre-existing recording. All configuration parameters are optional, although a "no changes requested" fault will occur if there are no optional parameters present.

Parameter	Type	Comments
recordingId	Integer	The recording ID for the recording to modify. This should be the identifier as returned by recording.enumerate.
recordingName (optional)	String	The name for the recording.
numericId (optional)	String	The numeric ID used for this recording.
pin (optional)	String	The PIN for this recording.
registerWithSIPRegistrar (optional)	Boolean	Whether to register this recording with the sip registrar.
registerWithGatekeeper (optional)	Boolean	Whether to register this recording with the H.323 gatekeeper.
playbackEnabled (optional)	Boolean	Whether this recording has playback enabled.

recording.delete

This method deletes a recording from the VCR.

Parameter	Type	Comments
recordingId	Integer	The recording ID for the recording to delete. This should be the identifier as returned by recording.enumerate.

A "no such recording" fault is returned if the recording does not exist.

recording.enumerate

Enumerates the recordings stored on the VCR.

Parameter	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified above. If required, enter the value returned in enumerateID by the most recent recording.enumerate response. If it is omitted, a new enumeration is started.
enumerateFilter (optional)	String	A filter expression.

Valid expressions within the enumerate filter are:

Expression	Type	Comments
recordingId	Integer	The unique index of this recording
internal	Boolean	True if the recording is stored internally
external	Boolean	True if the recording is stored externally
inProgress	Boolean	True if the recording is in the process of being made

This returns the following:

Response	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified above. The value which should be used in the next recording.enumerate call to get the next set of data. If this is omitted, no further data is available from the VCR.
recordings	Array	See below for details.

The "recordings" array contains the following structures:

Field	Type	Comments
recordingName	String	The name of this recording.
recordingId	Integer	A unique identifier for this recording.
folderId	Integer	The unique identifier of the folder in which this recording is stored.
external	Boolean	True if the recording is stored externally.

Field	Type	Comments
numericId	String	The numeric ID registered with the SIP registrar or H.323 gatekeeper.
pin	String	The PIN of this recording.
status	String	The current status of the recording. This can be any of the following: **`idle`**, **`initialising`**, **`invalid`**, **`uploading`**, **`deleting`** or **`recording`**.
playbackEnabled	Boolean	True if the recording can be played back by users.
registerWithSIPRegistrar	Boolean	True if the numeric ID is registered with the SIP registrar.
registerWithGatekeeper	Boolean	True if the numeric ID is registered with the H.323 gatekeeper.

recording.stop

This method stops a recording in progress on the VCR software version 2.2(1.15) and later. The connection between the VCR and the endpoint (or endpoints) involved in the call will be dropped.

Parameter	Type	Comments
recordingId	Integer	The recording ID for the recording to stop. This should be the recordingId as returned by recording.enumerate.

A "no such recording" fault is returned if the recording does not exist or is not in the process of being recorded; this method has an effect only on those recordings whose *status* value as returned by “recording.enumerate” is **recording**.

sip.query

Retrieves information about SIP configuration for an VCR. This function takes no parameters.

Response	Type	Comments
configuredRegistrar	String	The currently configured SIP registrar address. This corresponds to the “SIP registrar address” on the **Settings >SIP** web page, and will be an empty string value if there is no currently configured SIP registrar.
configuredProxy	String	The currently configured SIP proxy address. This corresponds to the “SIP proxy address” on the **Settings > SIP** web page, and will be an empty string value if there is no currently configured SIP proxy.
registrarContactURI	String	The URI used to register.
registrarContactDomain	String	The domain portion of the URI used to register.

Related information sources

system.xml

While not strictly part of the XML-RPC API, some information can be retrieved from the system.xml file. This can be downloaded via HTTP as the file system.xml in the root of the unit, for example http://vcr/system.xml.

An example is:

```
<system>
      <manufacturer>Cisco</manufacturer>
      <model>MSE 8220</model>
      <serial>SM010040</serial>
      <softwareVersion>2.4(1.2)</softwareVersion>
      <buildVersion>7.2(1.2)</buildVersion>
      <hostName>vcr</hostName>
      <maxVideoResolution>max</maxVideoResolution>
      <uptimeSeconds>939544</uptimeSeconds>
</system>
```

The meaning of the fields is:

Field	Comments
manufacturer	The manufacturer of this VCR, e.g. Cisco.
model	The model of this particular VCR.
serial	The serial number of this VCR.
softwareVersion	The software version currently running.
buildVersion	The build version of the software currently running.
hostName	The host name of the system.
uptimeSeconds	The number of seconds since boot.
maxVideoResolution	The maximum video resolution for the VCR; either "cif" or "max" if larger video is enabled.

Fault codes

The Cisco TelePresence VCRs, MCUs and gateways have a series of fault codes which are returned when a fault occurs during the processing of an XML-RPC request. While individual call descriptions above give some indication of which faults may occur, below is a description of all possible fault codes used within this specification and the most common interpretation. Note that not all codes are used by the VCR.

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.

Fault Code	Description
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	**Port limit lower than active**
26	**Floor control not enabled**
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.

A method of reducing the amount of TCP traffic when polling the VCR via the API is to use HTTP keep-alives. (This method can be used with other Cisco TelePresence products that also support the API, such as the MCU and ISDN gateway.)

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 Cisco product that the client supports HTTP keep-alives. The VCR *may* then choose to not close the TCP connection after returning its response to the request. If the connection will be closed, the VCR returns the following line in the HTTP header of its response:

```
Connection: close
```

The absence of this line indicates that the VCR will keep the TCP connection open and that the client may use the same connection for a subsequent request.
The VCR will not allow a connection to be kept alive if:

- the current connection has already serviced a set number of requests
- the current connection has already been open for a certain amount of time
- there are already more than a certain number of connections in a “kept alive” state

These restrictions are in place to limit the resources associated with kept-alive connections. If a connection is terminated for either of the first two reasons, the client will probably find that the connection is back in a keep-alive state following the next request.

The client should never assume a connection will be kept alive.

Also note that, even after a response not containing the “connection: close” header, the connection will still be closed if no further requests are made within one minute. If requests from the client are likely to be this far apart then there is little to be gained by using HTTP keep-alives.

References

The following table lists documents and web sites referenced in this document. All product documentation can be found on our web site.

[1]	XML-RPC, http://www.xmlrpc.com/
[2]	RFC 2616, http://www.faqs.org/rfcs/rfc2616.html