



Cisco TelePresence ISDN Gateway Remote Management API Reference Guide

D14659.04

March 2011

Contents

Contents	2
API version history	3
API version 2.7	3
API version 2.6	4
Introduction.....	5
HTTP and HTTPS	5
Consider API overhead when writing applications.....	5
Protocol overview	6
Message flow	6
Authentication	6
Example command message.....	6
Unicode support	9
HTTP headers.....	9
XML header.....	9
Common message elements	9
Authentication	9
Enumerate functions	10
Participant records	10
API reference.....	12
auditlog.delete	13
auditlog.query.....	13
calls.active.enumerate	14
calls.completed.enumerate	15
cdrlog.delete	17
cdrlog.enumerate	17
cdrlog.query	18
device.health.query	18
device.network.query	20
device.query	23
device.restartlog.query.....	24
feedbackReceiver.configure.....	25
Feedback events.....	26
feedbackReceiver.query	27
Feedback messages	28
Example feedback message.....	28
isdn.port.query	29
Related information sources.....	31
system.xml	31
Fault codes.....	32
HTTP keep-alives	34
References	35

API version history

The latest Cisco TelePresence ISDN Gateway Remote Management API is version 2.7. The following Cisco TelePresence ISDN Gateway products support this version, provided they are running software version 2.1 and later:

- ▶ ISDN GW 3200 Series
- ▶ ISDN GW 3241
- ▶ ISDN GW MSE 8310
- ▶ ISDN GW MSE 8321

This table specifies which Cisco TelePresence ISDN Gateway products support which API version:

API version	ISDN GW 3200 Series, 3241, MSE 8310, MSE 8321
2.4	Software version 1.3
2.5	Software version 1.4 and later
2.6	Software version 2.0 and later
2.7	Software version 2.1 and later

API version 2.7

Version 2.7 introduces the following changes:

XML-RPC request	Parameter	Addition/Deprecated
cdrlog.enumerate		Addition
feedbackReceiver.configure		Addition
feedbackReceiver.query		Addition
device.network.query	portA/portB structs: domainName hostName nameServer nameServerSecondary	Deprecated
	portA/portB structs: ipv4Enabled ipv6Enabled ipv6Conf ipv6Address ipv6PrefixLength defaultIpv6Gateway linkLocalIpv6Address linkLocalIpv6PrefixLength	Addition
	dns array: domainName hostName nameServer nameServerSecondary	Addition
system.xml		Addition

API version 2.6

Version 2.6 introduced the following changes:

XML-RPC request	Addition/Deprecated
auditlog.delete	Addition
auditlog.query	Addition
cdrlog.delete	Addition
cdrlog.query	Addition
device.query	Addition

Introduction

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

- ▶ ISDN GW 3200 Series
- ▶ ISDN GW 3241
- ▶ ISDN GW MSE 8310
- ▶ ISDN GW MSE 8321

This is accomplished via messages sent using the XML-RPC protocol. XML-RPC is a protocol for remote procedure calling using HTTP as the transport and XML as the encoding. It is designed to be as simple as possible but also to allow complex data structures to be transmitted, processed and returned.

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 31 characters.

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

HTTP and HTTPS

Cisco TelePresence ISDN Gateways 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 Cisco TelePresence ISDN Gateway products from software version 1.4 and later.

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

The Cisco TelePresence ISDN Gateway 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 ISDN gateway'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 ISDN gateway. 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 clients' requests and send API commands directly to the ISDN gateway. Implement some form of control in the API application on your server to prevent the ISDN gateway being overloaded with API commands. This provides better control than having clients send API commands directly and avoids impaired performance from unmanageable numbers of API requests.

Furthermore, the API is designed to have as little impact as possible on the network when responding to requests. The gateway's responses do not routinely include data that is not relevant, or empty data structures where the data is not available. Your application should take responsibility for checking whether the response includes what you expected, and you should design it to gracefully handle any situations where the device does not respond with the expected data.

XML-RPC protocol overview

Message flow

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

Authentication

To manage the Cisco TelePresence ISDN Gateway, the controlling application must authenticate itself as a user with relevant privileges. Accordingly, each message contains a user name and password (for details see the Authentication elements section below).



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

All calls require administrator privileges.

Example command message

Command messages are sent in XML format. For example, the following message queries port 7 on a Cisco TelePresence ISDN Gateway:

```
POST /RPC2 HTTP/1.1
User-Agent: Frontier/5.1.2 (WinNT)
Host: 10.2.1.100
Content-Type: text/xml
Content-length: 402

<?xml version="1.0"?>
<methodCall>
<methodName>isdn.port.query</methodName>
<params>
<param>
<value><struct>
<member>
<name>authenticationPassword</name>
<value><string></string></value>
</member>
<member>
<name>port</name>
<value><int>7</int></value>
</member>
<member>
<name>authenticationUser</name>
<value><string>admin</string></value>
</member>
</struct></value>
</param>
</params>
</methodCall>
```

If the command was successful, the ISDN gateway sends a success response. Note that in the following example some lines have been omitted at the ellipsis (...):

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

<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value><struct>
<member>
<name>layer2</name>
<value><boolean>1</boolean></value>
</member>
<member>
<name>layer1</name>
<value><boolean>1</boolean></value>
</member>
<member>
<name>searchHighLow</name>
<value><boolean>0</boolean></value>
</member>
<member>
<name>enabled</name>
<value><boolean>1</boolean></value>
</member>
<member>
<name>lowChannel</name>
<value><int>1</int></value>
</member>
<member>
<name>mode</name>
<value><string>terminal</string></value>
</member>
<member>
<name>directoryNumber</name>
<value><string></string></value>
</member>
<member>
<name>bChannels</name>
<value><array><data>
<value><struct>
<member>
<name>incoming</name>
<value><boolean>1</boolean></value>
</member>
<member>
<name>calling</name>
<value><string></string></value>
</member>
<member>
<name>active</name>
```

```

<value><boolean>1</boolean></value>
</member>
<member>
<name>voice</name>
<value><boolean>0</boolean></value>
</member>
<member>
<name>called</name>
<value><string>208201</string></value>
</member>
<member>
<name>channel</name>
<value><int>1</int></value>
</member>
</struct></value>
...
<value><struct>
<member>
<name>active</name>
<value><boolean>0</boolean></value>
</member>
<member>
<name>channel</name>
<value><int>31</int></value>
</member>
</struct></value>
</data></array></value>
</member>
<member>
<name>type</name>
<value><string>e1</string></value>
</member>
<member>
<name>port</name>
<value><int>7</int></value>
</member>
<member>
<name>highChannel</name>
<value><int>31</int></value>
</member>
</struct></value>
</param>
</params>
</methodResponse>

```

If the command fails (for example, querying port 7 on a 4-port ISDN gateway) the gateway sends a fault response:

```

HTTP/1.1 200 OK
Connection: close
Content-Type: text/xml
Content-Length: 411

<?xml version='1.0'?>
<methodResponse>
<fault>
<value><struct>

```

```

<member>
<name>faultCode</name>
<value><int>24</int></value>
</member>
<member>
<name>faultString</name>
<value><string>no such port</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 (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 (such as operationScope) and no enumerateID parameter.
2. The ISDN gateway returns with an array containing the requested data and possibly a new enumerateID.
3. If an enumerateID exists, the client calls the enumerate method again with any required/desired parameters and with an enumerateID parameter that contains the ID returned by the ISDN gateway from the previous call. This should be repeated while the ISDN gateway continues to provide new enumerateID values in responses.
4. After all data is returned the ISDN gateway will reply with all remaining results, but no enumerateID.

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

Participant records

Several functions return participant records, which have the following fields:

Field	Type	Comments
uniqueId	Integer	A unique index for this participant.
protocol	String	Either h323 (for IP) or h320 (for ISDN).
number	String (length <=64 in r1.4 & <=128 in r1.5)	The E164 number, IP address or DNS name of the participant. Empty if the call is a leased line call (as no number will be used in that mode).
name	String (length <128)	The name of this participant (e.g. the h323 name).
autoAttendant	boolean	True if this participant is an auto attendant running on the gateway, false otherwise.
incoming	boolean	True if the call is incoming to the gateway, false if the call is outgoing.
videoCodec	String	The video codec used for this participant.
audioCodec	String	The audio codec used for this participant.
progress	String	The state of the connection to this participant. One of: none, initial, proceeding, alerting, connected or finished. Only present for active participants.
fecc	boolean	True if far end camera control is established, false otherwise. Only present for active participants.
ipAddress	String	The IP address of the participant. Only present for IP participants. If the endpoint is connected through a gatekeeper in routed mode, then the address will be the IP address of the gatekeeper.

Field	Type	Comments
callIdentifier	base64	The h323 Call Identifier for this participant. Only present for h323 participants.
channelCount	Integer	The number of ISDN channels in use. Only present for ISDN participants.
channels	Array	An array of integers. The channels in use by this call. Only present for ISDN participants.
calledNumbers	Array	An array of strings. The ISDN number called on each of the channels in use by this call.

API reference

This section provides the following reference detail for each Cisco TelePresence ISDN Gateway API call:

- ▶ Call function description
- ▶ Parameter list
- ▶ Responses list
- ▶ Data types
- ▶ Structure formats
- ▶ Any additional information that may assist in using the API call
- ▶ Any parameters deprecated from previous versions

The ISDN gateway API calls are listed below. Click any entry to go to the reference page for that call.

API call	Page
auditlog.delete	13
auditlog.query	13
calls.active.enumerate	14
calls.completed.enumerate	15
cdrlog.delete	17
cdrlog.enumerate	17
cdrlog.query	18
device.health.query	18
device.network.query	20
device.query	23
device.restartlog.query	24
feedbackReceiver.configure	25
feedbackReceiver.query	27
isdn.port.query	29

auditlog.delete

Deletes stored events from the CDR log.

Parameter	Type	Comments
deleteIndex	Integer	You can delete logs in chunks of 400 entries. To delete entries, you can enter the value returned after an auditlog.query call in deleteableIndex. This will delete all complete chunks (400 entries) below this value, leaving the residuals. Alternatively, you can delete fewer entries by entering a number below the value of deleteableIndex. This will delete all complete chunks (400 entries) below the entered number, leaving any residuals

Stored audit events up to and including the indicated deleteIndex will be permanently deleted.

auditlog.query

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

Response	Type	Comments
firstIndex	Integer	The index of the oldest stored audit event.
deleteableIndex	Integer	The index of the most recent deletable audit event.
numEvents	Integer	The total number of events stored.
percentageCapacity	Integer	The percentage of total available capacity used by the audit log.

calls.active.enumerate

Returns a list of all currently active calls on the ISDN gateway.

Parameter	Type	Comments
enumerateID	Integer	Optional. An enumerateID, as specified in Enumerate functions .

This returns the following XML-RPC structure:

Response	Type	Comments
enumerateID	Integer	Optional. An enumerateID, as specified in Enumerate functions .
calls	Array	See below for details.

The calls structure contains the following fields:

Field	Type	Comments
uniqueId	Integer	A unique identifier for this call.
participantOne	Struct	Participant identification structures, as specified in Participant records .
participantTwo	Struct	
startTime	dateTime.iso8601	The start time of the call.
voiceCall	Boolean	True if this is a voice-only call, false for a video call.
aggregationCall	Boolean	True if this is an aggregation call, false otherwise.
callProgress	String	The state of the call. One of: initial, callingOut, connected or dying.
encryption	String	Either all, some or none, depending on the current encryption state of the media channels (on the IP side of the call).
ISDN encryption	String	Either all, some or none, depending on the current encryption state of the media channels on the ISDN side of the call.
maxDuration	Integer	The maximum duration of this call in seconds. If there is no maximum, this value is 0.
calledNumber	String (length <=64 in r1.4, <=128 in r1.5)	The number originally called, or unknown if this number is unknown. (The maximum string length changed from 64 in Cisco TelePresence ISDN Gateway software version 1.4 to 128 in software version 1.5.)
callDuration	Integer	The duration of the call in seconds.

Field	Type	Comments
callBandwidth	Integer	The bandwidth of the call in bits per second.

calls.completed.enumerate

Returns completed call information available in local memory. The information returned is the equivalent to the information on **Status** page and is limited to the last 100 calls.

This function takes no parameters.

Parameter	Type	Comments
enumerateID	Integer	Optional. An enumerateID, as specified in Enumerate functions .

This returns the following XML-RPC structure:

Response	Type	Comments
enumerateID	Integer	Optional. An enumerateID, as specified in Enumerate functions .
calls	Array	See below for details.

The array "calls" is a structure with the following fields:

Field	Type	Comments
uniqueId	Integer	A unique identifier for this call.
participantOne	Struct	Participant identification structures, as specified in Participant records .
participantTwo	Struct	
startTime	dateTime.iso8601	The start time of the call
endTime	dateTime.iso8601	The end time of the call.
voiceCall	Boolean	True if this is a voice-only call, false for a video call.
aggregationCall	Boolean	True if this is an aggregation call, false otherwise.
encryption	String	Either all, some or none, depending on the current encryption state of the media channels (on the IP side of the call).
ISDN encryption	String	Either all, some or none depending on the current encryption state of the media channels on the ISDN side of the call.
maxDuration	Integer	The maximum duration of this call in seconds. If there is no maximum, this value is 0.
calledNumber	String	The number originally called, or unknown if this number is unknown.

Field	Type	Comments
callBandwidth	Integer	The bandwidth of the call in bits per second.

cdrlog.delete

Deletes stored events from the CDR log.

Parameter	Type	Comments
deleteIndex	Integer	You can delete logs in chunks of 400 entries. To delete entries, you can enter the value returned after a cdrlog.query call in deleteableIndex. This will delete all complete chunks (400 entries) below this value, leaving the residuals. Alternatively, you can delete fewer entries by entering a number below the value of deleteableIndex. This will delete all complete chunks (400 entries) below the entered number, leaving any residuals

Stored audit events up to and including the indicated deleteIndex will be permanently deleted.

cdrlog.enumerate

Allows the calling application to download recent CDR log data. The call returns a subset of the CDR log based on filter, index and numEvents parameters:

Parameter	Type	Comments
filter	array	Optional. List of event types you are interested in. If omitted, all event types are returned. For the ISDN gateway, the call can request any or all of the following event types: <ul style="list-style-type: none"> ▶ newConnection ▶ connectionProceeding ▶ connectionFinished ▶ multiwayCallTransfer
index	int	Optional. Specifies the index from which to get events. The next index is returned by the target device's response (nextIndex). If this parameter is omitted (or is a negative number) then events are returned from the beginning of the log.
numEvents	int	Optional. Specifies the maximum number of events to be returned per enumeration. If this parameter is omitted (or is outside the range 1-20 inclusive) then a maximum of 20 events will be returned per enumeration.

The response returns reference information (time and log position) and an associative array (“events”) of event types to the following structure:

Parameter	Type	Comments
startIndex	Integer	The revision number provided (or if less than the target device has a record of, the first record the device does know about). Comparing this with the index provided gives the number of dropped logs.
nextIndex	Integer	The revision number of the data being provided. Reusable in a subsequent call to the API.
eventsRemaining	Boolean	Indicates whether there is data remaining after this. Provided to avoid putting all data in a single call.
events	Array	List of the new events. These are structures with some common fields (time, type and index) and other fields that are specific to the event type. All requested event types appear in the array, regardless of whether they actually exist or have data attached to them.
currentTime	dateTime. iso8601	Current time according to the target device. Provided to help establish the actual time of events in the client’s local time.

cdrlog.query

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

Field	Type	Comments
firstIndex	Integer	The index of the oldest stored CDR event.
deleteableIndex	Integer	The index of the most recent deletable CDR event.
numEvents	Integer	Total number of events stored.
percentageCapacity	Integer	The percentage of total available capacity used by the CDR log.

device.health.query

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

Response	Type	Comments
----------	------	----------

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	
rtcBatteryStatus	String	
rtcBatteryStatusWorst	String	
voltagesStatus	String	
voltagesStatusWorst	String	
operationalStatus	String	One of active, shuttingDown or shutDown.

device.network.query

This call takes no parameters and returns the following XML-RPC structures:

Parameter	Type	Comments
portA	Struct (see following table)	Contains the configuration and status for port A.
portB	Struct (see following table)	Contains the configuration and status for port B.
dns	Array (see dns array table)	Contains DNS parameters.

The portA and portB structs are as follows:

Field	Type	Comments
enabled	Boolean	True if the port is enabled, false otherwise.
ipv4Enabled	Boolean	True if IPv4 interface is enabled, false otherwise. (Not returned if the interface is disabled with no configured address for either IPv4 or IPv6.)
ipv6Enabled	Boolean	True if IPv6 interface is enabled, false otherwise. (Not returned if the interface is disabled with no configured address for either IPv4 or IPv6.)
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 duplex enabled.
macAddress	String	A 12-character string; no separators.
packetsSent	Integer	Stats from the web interface. Note that these values may wrap as they are 32 bit signed integers.
packetsReceived	Integer	
multicastPacketsSent	Integer	
multicastPacketsReceived	Integer	

Field	Type	Comments
bytesSent	Integer	
bytesReceived	Integer	
queueDrops	Integer	
collisions	Integer	
transmitErrors	Integer	
receiveErrors	Integer	
bytesSent64	String	64 bit versions of the above stats, using a string rather than an integer.
bytesReceived64	String	
Optional parameters		
hostName	String	Host name of the system. Deprecated.
dhcp	Boolean	True if IPv4 address configured by DHCP, false otherwise.
ipAddress	String	IPv4 address.
subnetMask	String	IPv4 subnet mask.
defaultGateway	String	Default gateway IPv4 address.
domainName	String	Domain name of the system. Deprecated.
nameServer	String	IPv4 address. Deprecated.
nameServerSecondary	String	IPv4 address. Deprecated.
ipv6Address	String	IPv6 address.
ipv6PrefixLength	Integer	IPv6 address prefix length.
defaultIpv6Gateway	String	Default gateway IPv6 address.
linkLocalIpv6Address	String	Link-local IPv6 address.
linkLocalIpv6PrefixLength	Integer	Link-local IPv6 address prefix length.

Note: Optional fields are returned only if the interface has been both enabled and configured.

The dns array is as follows:

Parameter	Type	Comments
hostName	String	Host name of the system.
nameServer	String	IPv4 or IPv6 address.
nameServerSecondary	String	IPv4 or IPv6 address.
domainName	String	Domain name of the system (DNS suffix).

device.query

This method call takes no parameters. 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 ISDN gateway
softwareVersion	String	The software version of the running software.
buildVersion	String	The build version of the running software.
model	String	The model of this ISDN gateway.
apiVersion	String	The version number of the API implemented by this ISDN gateway.
activatedFeatures	Array	Currently only contains a string "feature" with a short description of the feature.
isdnPorts	Integer	The number of ISDN port licenses available on the gateway.

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 is as follows:

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).

feedbackReceiver.configure

This call is used by a management server application (such as a Cisco TelePresence Management Server) to register as a feedback receiver when it first connects to the ISDN gateway. After registering, the management server application will listen for event notifications (“feedback messages”) from the ISDN gateway in response to certain, pre-configured changes (“feedback events”) on the gateway.

The management server application need not constantly poll the gateway to monitor activity. Instead the gateway automatically publishes feedback events as they occur, via feedback messages sent to the management server application in the form of HTTP POST or HTTPS POST requests. These messages can be used to prompt the management server to take a particular action.

Not all state or configuration changes on the ISDN gateway can be configured and published as feedback events. Only the following changes are supported as feedback events:

- ▶ restart
- ▶ configureAck
- ▶ connectionFinished

For more details, refer to [Feedback events](#) on page 26.

Parameter	Type	Comments
receiverURI	String	<p>A URI that identifies the management server application and the protocol. For example, <code>http://tms1:8080/RPC2</code></p> <p>If this parameter is absent or set to an empty string then the feedback receiver is effectively deconfigured and will receive no further notifications.</p> <p>Valid protocol types are HTTP and HTTPS. If no port number is specified then the protocol defaults are used (80 and 443 respectively).</p>
receiverIndex	Integer	<p>An integer in the range 1-20, inclusive. Determines which position (or “slot”) this receiver should use in the management server application’s feedback receivers table.</p> <p>If absent, slot 1 is assumed.</p> <p>If < 0, then any available slot is used.</p>
events	Struct	<p>An associative array that maps an event name string to a boolean expression (0 or 1). The string is one of <code>configureAck</code>, <code>restart</code> or <code>connectionFinished</code>. The boolean indicates whether the feedback receiver should receive an event notification message when that event occurs.</p> <p>If this parameter is absent the receiver is configured to receive all event notifications.</p>

sourceIdentifier	String	<p>Optional.</p> <p>If supplied, the identifier is returned in feedback messages (event notifications and feedbackreceiver.query responses) that relate to this receiver.</p> <p>If absent, the Ethernet port A MAC address of the ISDN gateway is used.</p>
------------------	--------	--

Feedback events

The feedback events that the ISDN gateway can publish are as follows:

Event	Description	Usage
restart	Sent at startup time.	
configureAck	Sent when an application successfully configures or reconfigures a feedback receiver.	
connectionFinished	Sent when a call has finished on the ISDN gateway.	<p>Prompts the management server application to retrieve recent call history details for the gateway.</p> <p>An example of intended use for the connectionFinished feedback event is as follows:</p> <ol style="list-style-type: none"> 1. Register for the event. 2. Wait for a feedback message with notification of the event. 3. Use the cdrlog.enumerate call to find out more information about the call that has finished.

feedbackReceiver.query

This method call takes no parameters and returns a Receivers array with details of all existing feedback receivers configured on the ISDN gateway. The Receivers array is as follows:

Parameter	Type	Comments
receiverURI	String	<p>A URI that identifies the management server application and the protocol. For example, <code>http://tms1:8080/RPC2</code></p> <p>If this parameter is absent or set to an empty string then the feedback receiver is effectively deconfigured and will receive no further notifications.</p> <p>Valid protocol types are HTTP and HTTPS. If no port number is specified the protocol defaults are used (80 and 443 respectively).</p>
sourceIdentifier	String	<p>A string that is returned in feedback notification events to identify the originator. The management server application allocates the identifier when configuring the feedback receiver, unless the default configuration option was used. If the default was used then the identifier is the Ethernet port A MAC address of the ISDN gateway.</p>
index	Integer	<p>The slot that this receiver uses in the management server application's feedback receivers table. A number between 1 and 20, inclusive.</p>

Feedback messages

Assuming that the transport protocol specified is HTTP or HTTPS, feedback messages follow the format used by the ISDN gateway for XML-RPC responses and do not contain cookie information or authentication. They are sent as HTTP POST or HTTPS POST requests to the specified URI. An example of the XML code for a feedback message is shown below.

Feedback messages contain two parameters:

- ▶ *sourceIdentifier* is a string that identifies the gateway, which may have been set by `feedbackReceiver.configure` or otherwise will be the gateway'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>
```

isdn.port.query

Returns the current status and settings of an ISDN port. (The device.query call returns the number of port licenses available on the ISDN gateway.)

Parameter	Type	Comments
port	Integer	The port number to query. This is zero based, so if there are four ports, they are numbered 0 to 3.

This function returns the following XML-RPC structure:

Response	Type	Comments
port	Integer	The port number.
type	String	The interface type. One of e1 , j1 , t1 or unknown .
mode	String	The interface mode. One of terminal , network or unknown .
layer1	Boolean	True if layer 1 is up, false otherwise.
layer2	Boolean	True if layer 2 is up, false otherwise.
enabled	Boolean	True if this port has been enabled.
bChannels	Array	Only present if layer 2 is up. See the bChannels structure description below for details.
lowChannel	Integer	The index of the low channel.
highChannel	Integer	The index of the high channel.
searchHighLow	Boolean	True if the search order is high to low, false if the search order is low to high.
directoryNumber	String	The directory number of this port.

The bChannels structure has the following members:

Field	Type	Comments
id	Integer	The channel index.
active	Boolean	True if this channel is active.
voice	Boolean	True if this is a voice call, false if a data call. Only present if active.

Field	Type	Comments
incoming	Boolean	True if this call is incoming, false if outgoing. Only present if active.
calling	String (length <64)	Only present if active.
called	String (length <64)	Only present if active.

This function will return a "No such port" fault (24) if the port requested does not exist.

Related information sources

system.xml

Although it is 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://ISDNGW/system.xml`).

An example `system.xml` file is as follows:

```
<system>
  <manufacturer>Codian</manufacturer>
  <model>ISDN GW 3241</model>
  <serial>SM000C00</serial>
  <softwareVersion>2.1(1.12)P</softwareVersion>
  <buildVersion>B.4.8(1.12)P</buildVersion>
  <hostName>ISDNGW</hostName>
  <isdnPorts>4</isdnPorts>
  <uptimeSeconds>14501</uptimeSeconds>
</system>
```

The fields in the `system.xml` file are as follows:

Field	Comments
manufacturer	The manufacturer of this ISDN gateway.
model	The model of this particular ISDN gateway.
serial	The serial number of this ISDN gateway.
softwareVersion	The software version that is currently running.
buildVersion	The build version of the software that is currently running.
hostName	The host name of the system.
isdnPorts	The number of ISDN ports.
uptimeSeconds	The number of seconds since boot.

Fault codes

The Cisco TelePresence gateways, MCU and VCRs have a series of fault codes which are returned when a fault occurs during the processing of an XML-RPC request.

This section describes all the fault codes used within this specification and their most common interpretation. Note that not all codes are used by the Cisco TelePresence ISDN Gateway.

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. Too many participants are already 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 specified PIN is not a valid series of digits.
14	Authorization failed. This code may be returned for a failed login attempt, in which case the supplied username or password, or both, may be incorrect.
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

Fault Code	Description
	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.
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 - <i>parameter_name</i> ".
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 - <i>parameter_name</i> ".
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 - <i>parameter_name</i> ".
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.

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

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 ISDN gateway *may* then choose not to close the TCP connection after returning its response to the request. If the connection will be closed, the ISDN gateway returns the following line in the HTTP header of its response:

```
Connection: close
```

The absence of this line indicates that the ISDN gateway will keep the TCP connection open and that the client may use the same connection for a subsequent request.

The ISDN gateway 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 that a connection will be kept alive.

Note: Even after a response that does not contain 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

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.