Cisco TelePresence ISDN Gateway API 2.10

Product Programming Reference Guide

D14964.02

March 2013

Contents

Introduction

This guide accompanies Version 2.10 of the Cisco TelePresence ISDN Gateway Remote Management API. The following Cisco TelePresence ISDN Gateway products support this version of the API provided that they are running software Version 2.2 or later:

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

Terminology

For clarity this guide uses the following conventions:

Term	Meaning
API	The remote management API for the Cisco TelePresence ISDN Gateway products.
Device	The specific model of the Cisco TelePresence ISDN Gateway that you are using.
Application	The calling application that sends API commands to the device.

API version history

This table lists the available API versions, and indicates which software versions of the Cisco TelePresence ISDN Gateway devices support which API versions:

API	Device software
2.10 (this version)	2.2 and later
2.7	2.1 and later
2.6	2.0 and later
2.5	1.4 and later
2.4	1.3 and later

Changes in Version 2.10

XML-RPC request	Parameter	Change
dialplan.modify		Added
dialplan.query		Added
dialplan.rule.add		Added
dialplan.rule.delete		Added
dialplan.rule.modify		Added
dialplan.rule.query		Added

XML-RPC request	Parameter	Change
dialplan.resetcounter		Added
dialplan.test		Added
gatekeeper.query		Added
isdn.settings.modify		Added
isdn.settings.query		Added
sip.query		Added
isdn.port.query	■ internationalPrefix ■ nationalPrefix ■ overlapReceivingLength	Added

Changes in Version 2.7

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

Changes in Version 2.6

XML-RPC request	Change
auditlog.delete	Added
auditlog.query	Added
cdrlog.delete	Added
cdrlog.query	Added
device.query	Added

API implementation

XML-RPC

The API is implemented as messages sent using XML-RPC, which is a simple protocol for remote procedure calling that uses HTTP (or HTTPS) as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible while allowing for complex data structures to be transmitted, processed, and returned. It has no platform or software dependence and was chosen in favor of SOAP (Simple Object Access Protocol) because of its simplicity.

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

The API implements all parameters and returned data as **<struct>** elements, each of which is explicitly named. For example, the **device.query** call returns the current time as a structure member named **currentTime** rather than as a single **<dateTime.iso8601>** value:

```
<member>
...<name>currentTime</name>
...<value><dateTime.iso8601>20050218T10:45:00</dateTime.iso8601></value>
</member>
```

Note: Unless otherwise stated in this guide, all strings have a maximum length of 31 characters.

For more information about XML-RPC see the XML-RPC specification.

Transport

The device implements HTTP/1.1 as defined by RFC 2616. It expects to receive HTTP communications over TCP/IP connections to port 80. The application should send HTTP POST messages to the URL defined by path **/RPC2** on the device IP address.

The device also supports HTTPS provided that it is running software Version 1.4 or later.

By default HTTPS is provided on TCP port 443. Optionally you can configure the device to receive HTTP and HTTPS connections on non-standard TCP port numbers.

Consider API overhead when writing applications

Every API command that your application sends incurs a processing overhead in the device's own application. The level of overhead varies with the type of command and the parameters sent.

If the device receives a high number of API commands every second, its performance could be seriously impaired (in the same way as if multiple users simultaneously accessed the device via the web interface). It is important to consider this overhead when designing your application architecture and software. The Design considerations [p.15] section provides recommendations for minimizing API overhead.

API messaging overview

The calling application can send XML command messages (calls) to the Cisco TelePresence ISDN Gateway (see the API reference section for details of the available calls and their associated parameters and responses).

Authentication

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

To manage the Cisco TelePresence ISDN Gateway, the controlling application must authenticate itself on the device as a user with relevant privileges. As the interface is stateless, it follows that every message must provide appropriate authentication. Depending on the certificate authentication options configured for the Cisco TelePresence ISDN Gateway, this can be provided in authentication parameters or by presenting a valid client certificate (see Common message elements [p.13] for details).

All calls require administrator privileges.

Message flow

The API message flow is as follows:

1. The application initiates the communication.
2. For each command sent (provided that the message is correctly formatted according to the XML-RPC specification) the device responds with a message that indicates success or failure.
   The response message may also contain any data that was requested. In the case of an error condition, the response may include only a fault code.

Examples of command and response messages are provided in API message examples [p.10]. Associated fault codes are listed in the Fault codes [p.55] section.

Encoding (ASCII or Unicode)

Your application can encode messages as ASCII text or as UTF-8 Unicode. If no method is specified, the API assumes ASCII.

If you want to specify the encoding, you can do it in a number of ways:

- Specifying encoding with HTTP headers
  Use the "Accept-Encoding: utf-8" header *or* modify the Content-Type header to read "Content-Type: text/xml; charset=utf-8".
- Specifying encoding with XML header
  The <?xml> tag is required at the top of each XML file. The API will accept an additional encoding parameter with value UTF-8 for this tag. That is, <?xml version="1.0" encoding="UTF-8"?>.

API message examples

Example command

This is an example of a command message that queries port 7 on a Cisco TelePresence ISDN Gateway device.

```
<?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>
```

Example response

Assuming that the command was well formed and the device is responsive, the device will respond with an XML **`methodResponse`** message. This is an example of a response message, which returns the data requested by the previous example command.

```
<?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</intx></value>
  </member>
  <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>
```

Example fault response

If the command fails (for example, querying port 7 on a 4-port device) the device sends a fault response.

```
<?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>
```

Common message elements

Authentication parameters

Unless the device is configured to allow (or require) certificate-based login, all messages must contain a user name and password as follows:

Parameter	Type	Description
authenticationUser	String	The name of a user with sufficient privilege for the operation being performed. The name is case sensitive.
authenticationPassword	String	The password that corresponds with the given user. The API ignores this parameter if the user has no associated password. This behavior differs from the web interface, where the password entry field must be blank in such cases.

Note: All calls require administrator privileges.

Certificate-based authentication modes

Specific authentication rules apply to API messages if client certificate-based security is enabled on the device (**Client certificate security** setting on the **Network > SSL certificates** page).

Client certificate security option	API authentication rules
Not required	No effect on API.
Verify certificate	Messages must have valid username and password values (**`authenticationUser`** and **`authenticationPassword`** parameters). To successfully make an HTTPS connection, the messages must also contain a valid client certificate that was issued by an authority that the device trusts.
Certificate-based authentication allowed	If the common name in the client certificate matches a username in the device configuration file, the API request is allowed access with the privileges assigned to that username. Messages do not need username and password values, which are ignored if present. If the common name does not match a username, all messages must include valid username and password values.
Certificate-based authentication required	Any username and password fields in the messages are always ignored. If the common name in the client certificate matches a username in the device configuration file, the API request is logged in with the privileges assigned to that username. If the common name does not match a username, the API request is rejected.

Enumerate methods

Because enumerate methods can potentially return a large volume of data, 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 sends an enumerate call with any necessary parameters and without an **`enumerateID`** parameter.
2. The device returns an array containing the enumerated items, and possibly an **`enumerateID`**.
3. If an **`enumerateID`** exists the application should call the enumerate method again, supplying the **`enumerateID`** returned by the previous call (and any required/desired parameters).
4. The application should repeat this process until the response fails to include an **`enumerateID`**, which indicates the enumeration is complete.

Note: Do not supply your own **`enumerateID`** values. Use only the values returned by the device.

Participant records

Several functions return participant records, as follows:

Field	Type	Description
uniqueID	Integer	A unique index for this participant.
protocol	String	Either h323 or sip (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. For SIP-based participants, the SIP URI. Empty if the call is a leased line call (no number is used in that mode).
name	String (length <128)	The name of this participant (for example, the H.323 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.
callIdentifier	base64	Only present for H.323 calls. The Call Identifier of the H.323 call, in the format of a base64-encoded GUID (globally unique identifier).
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.

Design considerations

Minimizing API overhead

It is essential to design your application architecture and software so that the processing load on the device application is minimized. To do this we recommend that you do the following:

- Use a single server to run the API application and to send commands to the device.
- 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. Then use the server to manage the clients' requests and send API commands directly to the device.
- Implement some form of control in the API application on your server to prevent the device being overloaded with API requests.

These measures provide much more control than having the clients send API commands directly, and will prevent the device performance being impaired by unmanageable numbers of API requests.

Unavailable or irrelevant data

The API is designed to minimize impact on the network when responding to requests, and device responses do not routinely include either irrelevant data or empty data structures where the data is unavailable.

It follows that your application should take responsibility for checking whether a response includes the expected data, and should be designed for graceful handling of situations where the device does not respond with the expected data.

Using HTTP keep-alives

If you are using API Version 2.4 or later, your application can use HTTP keep-alives to reduce the amount of TCP traffic that results from constantly polling the device (see HTTP keep-alives [p.16]).

HTTP keep-alives

Note: This feature is available with API Version 2.4 and later.

Your application can use HTTP keep-alives to reduce the amount of TCP traffic that results from constantly polling the device.

Implementation

Any client that supports HTTP keep-alives may include the following line in the HTTP header of an API request:

```
Connection: Keep-Alive
```

This line indicates to the device that the client supports HTTP keep-alives. The device *may* then decide that it will maintain the TCP connection after it has responded to the request.

If the device decides that it will not maintain the connection after it has responded, the device returns the following line in the HTTP header of its response:

```
Connection: close
```

Subject to the limitations mentioned below, if this line is absent from the HTTP header of the response, the device will keep the TCP connection open and the client may use the same connection for a subsequent request.

Limitations

The device 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 set 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 maintained (back in a keep-alive state) after the next request.

Usage considerations

The client should never assume that a connection will be maintained.

Also, even after a response that does not contain the `Connection:close` header, the device will close an open connection if no further requests are made by the client within one minute. If requests from the client are likely to be this far apart, there is little to be gained by using HTTP keep-alives.

API reference

This section details the API calls that are supported on the Cisco TelePresence ISDN Gateway.

Note: All parameters in the following call descriptions are required unless otherwise stated.

auditlog.delete

This call deletes stored events from the audit log.

Input parameters

Parameter	Type	Description
deleteIndex	Integer	To delete log entries, use either of these methods (logs are deleted in blocks of 400 entries): ■ Enter the **`deleteableIndex`** value returned after a log query call. This deletes all complete blocks (400 entries) below this value, and leaves any residual entries. ■ To limit the delete operation to fewer entries, enter a desired number that is lower than the value of **`deleteableIndex`**. This deletes all complete blocks below the specified number, and leaves any residual entries.

Stored events up to and including the indicated **`deleteIndex`** value will be permanently deleted.

auditlog.query

This call queries for statistics about the audit log. The call takes no parameters.

Returned data

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

calls.active.enumerate

This call returns a list of all currently active calls on the device.

Input parameters

Parameter	Type	Description
enumerateID	Integer	Optional. An enumerateID, as specified in Enumerate methods [p.13].

Returned data

Response	Type	Description
enumerateID	Integer	Optional. An enumerateID, as specified in Enumerate methods [p.13].
calls	Array (see table below)	

The format for the calls array is as follows:

Field	Type	Description
uniqueId	Integer	A unique identifier for this call.
participantOne	Struct	Participant identification structures, as specified in Participant records [p.14].
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	Depends on the current encryption state of the media channels on the IP side of the call. One of all, some, or none.
isdnEncryption	String	Depends on the current encryption state of the media channels on the ISDN side of the call. One of all, some, or none.
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 that number is not known. 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.
callBandwidth	Integer	The bandwidth of the call in bits per second.

calls.completed.enumerate

This call returns completed call information available in local memory. The information returned is limited to the last 100 calls.

Input parameters

Parameter	Type	Description
enumerateID	Integer	Optional. An enumerateID, as specified in Enumerate methods [p.13].

Returned data

Response	Type	Description
enumerateID	Integer	Optional. An enumerateID, as specified in Enumerate methods [p.13].
calls	Array (see table below)	

The format for the calls array is as follows:

Field	Type	Description
uniqueId	Integer	A unique identifier for this call.
participantOne	Struct	Participant identification structures, as specified in Participant records [p.14].
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	Depends on the current encryption state of the media channels on the IP side of the call. One of all, some, or none.
isdnEncryption	String	Depends on the current encryption state of the media channels on the ISDN side of the call. One of all, some, or none.
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 that number is not known.
callBandwidth	Integer	The bandwidth of the call in bits per second.

cdrlog.delete

This call permanently deletes stored events from the CDR log.

Input parameters

Parameter	Type	Description
deleteIndex	Integer	To delete log entries, use either of these methods (logs are deleted in blocks of 400 entries): ■ Enter the **`deleteableIndex`** value returned after a log query call. This deletes all complete blocks (400 entries) below this value, and leaves any residual entries. ■ To limit the delete operation to fewer entries, enter a desired number that is lower than the value of **`deleteableIndex`**. This deletes all complete blocks below the specified number, and leaves any residual entries.

Stored events up to and including the indicated **`deleteIndex`** value will be permanently deleted.

cdrlog.enumerate

This call allows the calling application to download recent CDR log data. It returns a subset of the CDR log based on filter, index, and numEvents parameters.

Input parameters

Parameter	Type	Description
filter	array	Optional. List of event types you are interested in. If omitted, all event types are returned. The call can request any or all of the following event types: ■ newConnection ■ connectionProceeding ■ connectionFinished ■ multiwayCallTransfer
index	int	Optional. Specifies the index from which to get events. The next index is returned by the response of the target device (**`nextIndex`**). Events are returned from the beginning of the log if this parameter is omitted (or is a negative number).
numEvents	int	Optional. Specifies the maximum number of events to be returned per enumeration. If this parameter is omitted (or is outside range 1-20 inclusive) a maximum of 20 events will be returned per enumeration.

Returned data

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

Parameter	Type	Description
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 returns statistics about the CDR log. The call takes no parameters.

Returned data

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

device.health.query

This call returns the current status of the device, such as health monitors and CPU load. The call takes no parameters.

Note: Some values do not apply to all device types.

Returned data

Response	Type	Description
cpuLoad	Integer	The CPU load, as a percentage value.
fanStatus	String	One of ok, outOfSpec or critical.
fanStatusWorst	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 returns network information for the device. The call takes no parameters.

Returned data

Response	Type	Description
portA	Struct (see portA/portB table below)	Contains the configuration and status for port A.
portB	Struct (see portA/portB table below)	Contains the configuration and status for port B.
dns	Array (see dns array table below)	Contains DNS parameters.

The format for the portA and portB structs is as follows:

Field	Type	Description
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 Mb/s).
fullDuplex	Boolean	True if full duplex enabled, false if half duplex enabled.
macAddress	String	A 12-character string; no separators.
packetsSent	Integer	Statistics from the web interface. These values are 32-bit signed integers, so may wrap.
packetsReceived	Integer	
multicastPacketsSent	Integer	
multicastPacketsReceived	Integer	
bytesSent	Integer	
bytesReceived	Integer	
queueDrops	Integer	
collisions	Integer	
transmitErrors	Integer	
receiveErrors	Integer	
bytesSent64	String	64-bit versions of the above statistics, using a string rather than an integer.
bytesReceived64	String	
Optional fields		
hostName	String	The host name of the device. Deprecated in API Version 2.7 and later.
dhcp	Boolean	True if IPv4 address is configured by DHCP, false otherwise.

Field	Type	Description
ipAddress	String	IPv4 address.
subnetMask	String	IPv4 subnet mask.
defaultGateway	String	The default gateway IPv4 address.
domainName	String	The domain name of the device. Deprecated in API Version 2.7 and later.
nameServer	String	IPv4 address. Deprecated in API Version 2.7 and later.
nameServerSecondary	String	IPv4 address. Deprecated in API Version 2.7 and later.
ipv6Address	String	IPv6 address.
ipv6PrefixLength	Integer	IPv6 address prefix length.
defaultIpv6Gateway	String	The default gateway IPv6 address.
linkLocalIpv6Address	String	The link-local IPv6 address.
linkLocalIpv6PrefixLength	Integer	The link-local IPv6 address prefix length.

Optional fields are returned only if the interface is both enabled and configured.

The format for the dns array is as follows:

Parameter	Type	Description
hostName	String	The host name of the device.
nameServer	String	IPv4 or IPv6 address.
nameServerSecondary	String	IPv4 or IPv6 address.
domainName	String	The domain name of the device (DNS suffix).

device.query

This call returns high-level status information for the device. The call takes no parameters.

Returned data

Response	Type	Description
currentTime	dateTime.iso8601	The current system time (UTC format).
restartTime	dateTime.iso8601	The date and time at which the device was last booted.
serial	String	The serial number of the device.
softwareVersion	String	The software version of the running software.
buildVersion	String	The build version of the running software.
uptime	Integer	The number of seconds since the device booted.
model	String	The model type of this device.
apiVersion	String	The version number of the API implemented by this device.
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 device.

device.restartlog.query

This call returns the restart log (also known as the system log on the web interface). The call takes no parameters.

Returned data

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

The format for the log array is as follows:

Field	Type	Description
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.

dialplan.modify

This call reorders the rules in the specified dial plan, according to the ordering given in the call.

Input parameters

Parameter	Type	Description
dialPlanName	String	The dial plan to be modified. One of ip_to_isdn, isdn_to_ip, leased_line_to_ip, or ip_to_leased_line.
revision	Integer	The current dial plan revision number. The value must be greater than zero.
rules	Array of integers	The unique identifiers of existing rules in the desired order, starting with the highest priority. Each value must be greater than zero. The number of integers in the array must match the number of rules in the dial plan.

Returned data

Response	Type	Description
revision	Integer	The new dial plan revision number after the change.

dialplan.query

This call returns a list of rules and their ordering for the specified dial plan. (To query for detailed information about rules use the dialplan.rule.query [p.38] call.)

Input parameters

Parameter	Type	Description
dialPlanName	String	The dial plan to be queried. One of ip_to_isdn, isdn_to_ip, leased_line_to_ip, or ip_to_leased_line.

Returned data

Response	Type	Description
revision	Integer	Positive integer that represents the dial plan version (required for subsequent calls).
rules	Array of struct (see table below)	The rules that exist in this dial plan.

The format for the rules array is as follows:

Field	Type	Description
name	String	The name of the dial plan rule.
uid	Integer	The unique identifier of the dial plan rule.
index	Integer	The index of the rule in the dial plan, starting at 0 for the highest priority rule and ascending incrementally.

dialplan.rule.add

This call adds a new rule to the bottom of the specified dial plan.

Parameter requirements for this call

- Parameters that comprise a regular expression (including **`calledPattern`** and **`callingPattern`**) are limited to 200 characters. Note that the device may return a longer value because it adds spaces for readability purposes.
- Encryption-related parameters are always required, even if encryption is currently or always disabled for the device. (To verify the current encryption status of the device, use the **`device.query`** call—the **`activatedFeatures`** field indicates if encryption is active.)

Input parameters

Parameter	Type	Description
dialPlanName	String	The target dial plan for the new rule. One of ip_to_isdn, isdn_to_ip, leased_line_to_ip, or ip_to_leased_line.
revision	Integer	The *current* dial plan revision number. The value must be greater than zero.
rule	Struct (see table below)	The settings for the new rule.

The format for the rule struct is as follows:

Field	Type	Description
name	String	The name of the rule.
condition	Struct	The condition part of the rule. See the condition struct table below.
action	Struct	The action part of the rule. See the action struct table below.

The format for the condition struct is as follows:

Field	Type	Description
incomingCallType	String	Which types of call the rule will match. One of any, video, or telephone. This field does not apply in SIP-related rules and is ignored by the device in such cases. Not required for leased line to IP.
matchCalled	String	How to match on called number. One of any, none, or custom. Not required for leased line to IP.
calledPattern	String	The regular expression that is matched against the called number. Only required if **`matchCalled`** is *custom*.
matchCalling	String	How to match on the calling number. One of any, none, or custom. Not required for leased line to IP.

Field	Type	Description
callingPattern	String	The regular expression that is matched against the calling number. Only required if **`matchCalling`** is *custom*.
protocol	String	The incoming protocol on which to match. One of any, sip, or h323. Only required for incoming IP calls.
matchPort	Integer	The port on which to match incoming calls (0 means any port). Only required for leased line to IP. The value must be less than or equal to the number of ports on the device.
matchGroup	Integer	The leased line group on which to match incoming calls. Only required for leased line to IP.

The format for the action struct is as follows:

Field	Type	Description
callAction	String	The action to take when the rule is matched. One of reject, autoAttendant, call, autoAttendantTcs4, or leasedLineGroup. The leasedLineGroup field is valid only in leased line mode and autoAttendantTcs4 is valid only for calls to the IP network.
callProtocol	String	The protocol to use for the new call. One of h323 or sip. Only required for outgoing IP calls and not required in that case if the action is reject.
outgoingCallType	String	The type of call to make. For outgoing IP calls, one of telephone, matchIncoming, or video. For outgoing ISDN calls, one of bonding, aggregation, telephone, or matchIncoming. Not required for leased line, or when the action is reject or the call protocol is SIP.
outgoingTransport	String	The trunk type. One of trunk, udp, tcp, or tls. Only required when callProtocol is SIP.
useOriginalCalled	Boolean	Whether to use the original called number or a custom number. Not required for IP to leased line, or when the call type is aggregation or the action is reject, autoAttendant, or autoAttendantTcs4.
dialPattern	String	The pattern to dial. Only required if useOriginalCalled is false.
callManyPatterns	Array of strings	The patterns to dial for an aggregation call. Only required if the call type is aggregation, and not required in that case when the action is reject (never required for IP to leased line).
callPort	String	The port on which to make a call. Only required for IP to leased line, and not required in that case when the action is reject.
callGroup	String	The leased line group to call. Only required for IP to leased line, and not required in that case when the action is reject.
tcs4Digits	String	The TCS-4 digits to send. Only required for IP to leased line, and not required in that case when the action is reject.
useOriginalCalling	Boolean	Whether to use the original calling number or a custom one. Not required for IP to leased line or when the action is reject.

Field	Type	Description
newCallingPattern	String	The calling number to send in the new call. Only required if useOriginalCalling is false.
restrict	Boolean	Whether to make a restricted call. Not required when the action is reject, or the dial plan is towards IP, or the call type is telephone.
fallbackToTelephone	Boolean	Whether to fall back to a telephone call if video fails. Only required for IP to ISDN dial plans (not leased line) and not required in that case if the action is reject or the call type is telephone.
maximumChannelsAllowed	Integer	The maximum bandwidth, in B-channels. Valid values are -1, 1, 2, 3, 4, 5, 6, 8, 12, 18, 23, 24, or 30. Use -1 for the default setting. Not required if the action is reject or for IP to leased line. Values 24 and 30 are not valid in T1 or J1 mode.
ports	Array of integers	The ports on which the call can be made (<= number of ports on device). Not required if the action is reject or for leased line dial plans.
useTransparentEncryption	Boolean	Whether to use transparent encryption. Must be specified even if encryption is not activated on the device.
encryptionIp	String	Encryption setting for the IP side. One of required or optional. Must be specified even if encryption is not activated on the device.
encryptionIsdn	String	Encryption setting for the ISDN side. One of disabled, required, or optional. Must be specified even if encryption is not activated on the device. In the case of the IP to ISDN dial plan, if outgoingCallType is 'telephone', setting this parameter to required will cause a conflicting parameters error.
codecs	Struct	The codecs settings for the rule. Not required if the action is reject or the call type is telephone. See the codecs struct table below.

The format for the codecs struct is as follows:

Field	Type	Description
restriction	String	A description of the codec settings. One of default, custom, or safe.
audio	Struct	The permissible audio codecs. Only required if restriction is set to custom. See the audio struct table below.
video	Struct	The permissible video codecs. Only required if restriction is set to custom. See the video struct table below.
extendedVideo	Struct	The permissible video codecs for content. Only required if restriction is set to custom. See the extendedVideo struct table below.

The format for the audio struct is as follows:

Field	Type	Description
g722	Boolean	Whether the codec is allowed.

Field	Type	Description
g722_1	Boolean	Whether the codec is allowed (G.722.1).
g722_1c	Boolean	Whether the codec is allowed (G.722.1 Annex C).
g728	Boolean	Whether the codec is allowed.

The format for the video struct is as follows:

Field	Type	Description
h263	Boolean	Whether the codec is allowed.
h264	Boolean	Whether the codec is allowed.

The format for the extendedVideo struct is as follows:

Field	Type	Description
h263	Boolean	Whether the codec is allowed.
h264	Boolean	Whether the codec is allowed.

Returned data

Response	Type	Description
revision	Integer	The new dial plan revision number.
uid	Integer	The unique identifier of the dial plan rule that has been added.

dialplan.rule.delete

This call deletes a specific rule from a dial plan.

Input parameters

Parameter	Type	Description
dialPlanName	String	One of ip_to_isdn, isdn_to_ip, leased_line_to_ip, or ip_to_leased_line.
uid	Integer	The unique identifier of the dial plan rule to be deleted.
revision	Integer	The current dial plan revision number when the request is made.

Returned data

Response	Type	Description
revision	Integer	The new dial plan revision number after the change.

dialplan.rule.modify

This call modifies an individual rule in a dial plan with the values specified in the call.

Parameter requirements for this call

Unless explicitly stated otherwise, all parameters are mandatory. We recommend that you run a query call first and use the response as the basis for the modify call, by adjusting the settings you want to change.

- Parameters that comprise a regular expression (including **`calledPattern`** and **`callingPattern`**) are limited to 200 characters. Note that the device may return a longer value because it adds spaces for readability purposes.
- Encryption-related parameters are always required, even if encryption is currently or always disabled for the device. (To verify the current encryption status of the device, use the **`device.query`** call—the **`activatedFeatures`** field indicates if encryption is active.)

Input parameters

Parameter	Type	Description
dialPlanName	String	The dial plan to be modified. One of ip_to_isdn, isdn_to_ip, leased_line_to_ip, or ip_to_leased_line.
uid	Integer	The unique identifier of the dial plan rule.
revision	Integer	The current dial plan revision number when the request is made.
rule	Struct	The rule details, as described in dialplan.rule.add [p.32].

Returned data

Response	Type	Description
revision	Integer	The new dial plan revision number after the change.

dialplan.rule.query

This call returns detailed information about the specified dial plan rule, and a hit count of the rule usage.

Input parameters

Parameter	Type	Description
dialPlanName	String	The dial plan that contains the rule to be queried. One of ip_to_isdn, isdn_to_ip, leased_line_to_ip, or ip_to_leased_line.
uid	Integer	The unique identifier of the dial plan rule that is to be queried.

Returned data

Response	Type	Description
revision	Integer	The dial plan revision number when the request was made.
rule	Struct (see rule table below)	The rule details plus the rule ID and a usage hit count.

The format for the rule struct is as follows:

Field	Type	Description
Rule details as described in dialplan.rule.add [p.32].		
uid	Integer	The unique identifier of the dial plan rule.
hitCount	Integer	The number of times the rule has been matched since the device was last rebooted.

dialplan.test

This call evaluates the dial plan against the specified parameters.

Input parameters

Parameter	Type	Description
dialPlanName	String	The dial plan to be tested. One of ip_to_isdn, isdn_to_ip, leased_line_to_ip, or ip_to_leased_line.
revision	Integer	The current dial plan revision number. Value must be greater than zero.
protocol	String	The protocol for the incoming call. One of h323 or sip. Only required for IP to ISDN calls.
callingNumber	String	The number that is calling. Not required in leased line to IP.
calledNumber	String	The number that was called. Not required in leased line to IP.
callType	String	The type of call being made. One of video or telephone. Not required in leased line to IP or if the protocol is sip.
port	Integer	The port on which the call is coming in. Valid values are in the range 1 - *n*, where *n* is the number of ports on the device. Only required in leased line to IP.
leasedLineGroup	Integer	The leased line group on which the call is coming in.Valid values are in the range 1 - *n*, where *n* is the maximum number of leased line groups. Only required in leased line to IP.

Returned data

Response	Type	Description
revision	Integer	The new dial plan revision number.
outcome	String	One of reject, call, autoAttendant, autoAttendantTcs4, or callLeasedLineGroup. If a rule with a reject action is matched, the outcome is *reject* and the only other fields returned are **`revision`** and **`uid`**. If no rule is matched, the outcome is *reject* but only **`revision`** is present.
destination	String	[Non-aggregation calls only] The evaluated number to call. Not present in IP to leased line. In other cases present if **`outcome`** is *call*.
destinations	Array of strings	[Aggregation calls only] The evaluated numbers to call. Not present in IP to leased line. In other cases present if **`outcome`** is *call*.
callingNumber	String	The evaluated number from which to call. Not present in IP to leased line.
tcs4Digits	String	The evaluated TCS-4 number that will be sent. Only present in IP to leased line.
uid	Integer	The unique identifier of the dial plan rule that the condition matched.
callPort	Integer	The port on which the call will be made. Only present in IP to leased line.
callGroup	Integer	The leased line group to which the call will be made. Only present in IP to leased line.

dialplan.resetcounter

This call resets the rule hit counters for the specified dial plan to zero. (Note that all rule hit counters are automatically reset when the device is rebooted.)

Parameter	Type	Description
dialPlanName	String	The dial plan to be reset. One of ip_to_isdn, isdn_to_ip, leased_line_to_ip, or ip_to_leased_line.

The response returns the following:

Response	Type	Description
revision	Integer	The new dial plan revision number after the change.

feedbackReceiver.configure

This call is used by a management server application to register as a feedback receiver when it first connects to the device. The call configures the device to send feedback about the specified events to the specified receiver URI.

Input parameters

Parameter	Type	Description
receiverURI	String	A URI that identifies the management server application and the protocol (such as http://test1:8080/RPC2). Valid protocol types are HTTP and HTTPS. The protocol defaults are used if no port number is specified. The defaults are 80 and 443 respectively. If this parameter is absent or set to an empty string then the feedback receiver is effectively deconfigured and receives no further notifications.
receiverIndex	Integer	An integer in the range from 1 to 20 inclusive. Determines the position (slot) of this feedback receiver in the management server application's table of feedback receivers. If absent, slot 1 is assumed. If < 0, any available slot is used.
events	Struct	An associative array that maps an event name string to a boolean expression (0 or 1). The string is one of configureAck, restart, or connectionFinished. The boolean indicates whether the feedback receiver should receive an event notification message when that event occurs. If absent, the receiver is configured to receive all event notifications.
sourceIdentifier	String	Optional. If supplied, the identifier is returned in feedback messages (event notifications and feedbackreceiver.query responses) that relate to this receiver. If absent, the MAC address of the device is used.

feedbackReceiver.query

This call returns a list of all existing feedback receivers configured on the device. The call takes no parameters.

Returned data

Response	Type	Description
receivers	Array	Contains receivers in structures as described below.

The format of the receivers array is as follows:

Field	Type	Description
receiverURI	String	A URI that identifies the management server application and the protocol (such as http://test1:8080/RPC2). Valid protocol types are HTTP and HTTPS. The protocol defaults are used if no port number is specified. The defaults are 80 and 443 respectively.
sourceIdentifier	String	A string 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 MAC address of the device.
index	Integer	Identifies the slot that this receiver uses in the management server application's feedback receivers table. A number from 1 to 20 inclusive.

gatekeeper.query

This call retrieves the H.323 gatekeeper settings and current status. The call takes no parameters.

Returned data

Response	Type	Description
gatekeeperUsage	String	The usage state of the H.323 gatekeeper. One of enabled, disabled, or required. If disabled, no other values are returned.
address	String	The address of the H.323 gatekeeper. May be a host name or an IP address.
dnsStatus	String	The current status of the DNS lookup of the H.323 gatekeeper address. One of inProgress, resolved, or failed.
ip	String	The resolved IP address of the H.323 gatekeeper. Present only if **`dnsStatus`** is resolved.
activeRegistrations	Integer	The number of active registrations.
pendingRegistrations	Integer	The number of registrations in progress.
registrationType	String	Registration type (gateway or gatewayCisco).
portAssociationA	Boolean	True if interface 'PortA IPv4' is associated with the H.323 gatekeeper.
portAssociationB	Boolean	True if interface 'PortB IPv4' is associated with the H.323 gatekeeper.
portAssociationAv6	Boolean	True if interface 'PortA IPv6' is associated with the H.323 gatekeeper.
portAssociationBv6	Boolean	True if interface 'PortB IPv6' is associated with the H.323 gatekeeper.
h323ID	String	The H.323 ID used by the device to register with the H.323 gatekeeper.
deregisterIfNoLink	Boolean	Defines whether to deregister if the link is down.
sendResourceAvailabilityIndications	Boolean	Defines whether the device will send resource availability indications to the H.323 gatekeeper.
availabilityThresholdChannelsPercentage	Integer (<= 100)	The channel load percentage at which resource availability indications (RAIs) are sent to the H.323 gatekeeper. If channel availability is below this threshold the device does not send RAIs. Present only if the device is configured to send RAIs (**`sendResourceAvailabilityIndications`** is True).
dialPlanPrefixes	Array of strings	If set, up to ten prefixes that if dialed by a user will cause the H.323 gatekeeper to direct the call to the device.

Response	Type	Description
registeredAddress	String	The local signaling address and port (*[address]* : *[port]*). that the device has registered with the H.323 gatekeeper.
alternateGatekeepers	Integer	The number of alternate gatekeepers (if any) configured on the H.323 gatekeeper.
resourceAvailabilityStatus	String	Indicates the current resource availability for the device. One of available, unavailable, or disabled. If disabled, the device is not configured to send resource availability indications.
h323IDStatus	String	The current status of the H.323 ID registration process. One of idle, registering, registered, deregistering, waitingRetry, destroyPending, or unknown.

isdn.port.query

This call returns the current status and settings of the specified ISDN port (to query the number of port licenses on the device use the **`device.query`** call instead).

Input parameters

Parameter	Type	Description
port	Integer	The port number to query. Port numbers are zero-based (so in the case of a four-port device for example, the ports are numbered 0 to 3).

Returned data

Response	Type	Description
port	Integer	The port number. If the requested port does not exist, fault 24 (no such port) is returned.
type	String	The interface type. One of E1, T1, J1, 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 is enabled.
bChannels	Array	Only present if layer 2 is up. See the bChannels array table below.
lowChannels	Integer	The index of the low channel.
highChannels	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.
overlapReceivingLength	Integer (0-64)	The overlap receiving number length. 0 means disabled.
nationalPrefix	String (length <= 20)	Prefix for national calling party numbers.
internationalPrefix	String (length <= 20)	Prefix for international calling party numbers.

The format for the bChannels array is as follows:

Field	Type	Description
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.
incoming	Boolean	True if this call is incoming, false if outgoing. Only present if active.

Field	Type	Description
calling	String (length <64)	Only present if active.
called	String (length <64)	Only present if active.

isdn.settings.modify

This call modifies the ISDN settings for the device.

Unless explicitly stated otherwise, all parameters are mandatory. We recommend that you run a query call first and use the response as the basis for the modify call, by adjusting the settings you want to change.

Input parameters

Parameter	Type	Description
interfaceType	String	One of E1, T1, or J1.
switchType	String	One of national_isdn, 4ess, or dms100. This field is not present for E1 interfaces or on ISDN GW 3200 Series and ISDN GW MSE 8310 devices.
leasedLineMode	Boolean	
sendSendingComplete	Boolean	
legacyCapabilities	Boolean	
sendCallingNumberToIsdn	String	One of always, ifNumeric, or never.
maxIncomingCallRate	Integer	0,1,2,3,4,5,6,8,12,18,23,24, or 30. 0 means telephone. 24 through 30 apply only in E1 mode.
maxOutgoingCallRate	Integer	0,1,2,3,4,5,6,8,12,18,23,24, or 30. 0 means telephone. 24 through 30 apply only in E1 mode.
maxCallDuration	Integer	0,30,60,120,300,600,900,1800,3600,7200,18000,43200, or 82800. The maximum call duration in seconds. 0 means no limit.
allowParallelDialling	Boolean	
portSearchOrder	String	One of lowToHigh or highToLow.
loadBalancingActive	Boolean	
useHigherNumberedChannels	Boolean	This field is present only if load balancing is active. Defaults to True if not specified in the call.
outgoingCallsLayerTwo	String	One of establish or require.
audioCodecs	Struct	See the audioCodecs struct table below.
videoCodecs	Struct	See the videoCodecs struct table below.
contentCodecs	Struct	See the contentCodecs struct table below.
floorAndChairControlEnabled	Boolean	

The format for the audioCodecs struct is as follows:

Field	Type	Description
g722	Boolean	
g722_1	Boolean	G.722.1
g722_1c	Boolean	G.722.1 Annex C
g728	Boolean	

The format for the videoCodecs struct is as follows:

Field	Type	Description
h263	Boolean	
h264	Boolean	

The format for the contentCodecs struct is as follows:

Field	Type	Description
h263	Boolean	
h264	Boolean	

isdn.settings.query

This call retrieves the current ISDN settings for the device. The call takes no parameters.

Returned data

Response	Type	Description
restartRequired	Boolean	
interfaceType	String	One of E1, T1, or J1.
switchType	String	One of national_isdn, 4ess, or dms100. This field is not present on ISDN GW 3200 Series and ISDN GW MSE 8310 devices.
leasedLineMode	Boolean	
sendSendingComplete	Boolean	
legacyCapabilities	Boolean	
sendCallingNumberToIsdn	String	One of always, ifNumeric, or never.
maxIncomingCallRate	Integer	0,1,2,3,4,5,6,8,12,18,23,24, or 30. 0 means telephone. 24 through 30 apply only in E1 mode.
maxOutgoingCallRate	Integer	0,1,2,3,4,5,6,8,12,18,23,24, or 30. 0 means telephone. 24 through 30 apply only in E1 mode.
maxCallDuration	Integer	0,30,60,120,300,600,900,1800,3600,7200,18000,43200, or 82800. The maximum call duration in seconds. 0 means no limit.
allowParallelDialling	Boolean	
portSearchOrder	String	Either lowToHigh or highToLow.
loadBalancingActive	Boolean	
useHigherNumberedChannels	Boolean	This field is present only if load balancing is active.
outgoingCallsLayerTwo	String	One of establish or require.
audioCodecs	Struct	See the audioCodecs struct table below.
videoCodecs	Struct	See the videoCodecs struct table below.
contentCodecs	Struct	See the contentCodecs struct table below.
floorAndChairControlEnabled	Boolean	
advancedSettings	Struct	See the advancedSettings struct table below.

The format for the audioCodecs struct is as follows:

Field	Type	Description
g722	Boolean	
g722_1	Boolean	G.722.1
g722_1c	Boolean	G.722.1 Annex C
g728	Boolean	

The format for the videoCodecs struct is as follows:

Field	Type	Description
h263	Boolean	
h264	Boolean	

The format for the contentCodecs struct is as follows:

Field	Type	Description
h263	Boolean	
h264	Boolean	

The format for the advancedSettings struct is as follows:

Field	Type	Description
advertiseOutOfBandDtmf	Boolean	
specifyNationalNumberType	Boolean	
internationalPrefix	String	
crc4Enabled	Boolean	
esfEnabled	Boolean	
sendChannelId	Boolean	
isdnLineLength	String	Either short or long.
isdnLineImpedance	Integer	One of 75, 100, 110, or 120.
lineCoding	String	Either hdb3 or ami.
pulseShape	String	One of default, g703, or cs03.
nationalBits	String	Any 5-digit binary string.
videoNsf	Integer	A value from -1 to 31. -1 means none.
telephoneNsf	Integer	A value from -1 to 31. -1 means none.

sip.query

This call retrieves information about the SIP configuration on the device.

Returned data

Response	Type	Description
trunks	Array of trunk structs (see table below)	The array of trunk settings.

The format for the trunk structs is as follows:

Field	Type	Description
id	Integer (0)	The trunk ID. Currently only one trunk is supported, with ID '0'.
outboundAddress	String	The address of the proxy.
outboundDomain	String	The domain to append to outgoing requests.
outgoingTransport	String	The outgoing transport protocol of the SIP trunk. One of "udp", "tcp", or "tls".

Feedback receivers

The API allows you to register a management server application as a feedback receiver. After registering, the management server application will listen for event notifications from the device in response to pre-defined feedback events (changes) on the device.

The application does not need to constantly poll the device. Instead the device automatically publishes events when they occur, via feedback messages to the application in the form of HTTP POST or HTTPS POST requests. These messages can be used to prompt the application to take a particular action.

Not all state or configuration changes on the device can be published as feedback events. The events that the Cisco TelePresence ISDN Gateway can publish are listed in Feedback events [p.53] below.

Calls available

- Use feedbackReceiver.configure [p.41] to register a receiver to listen for one or more feedback events.
- Use feedbackReceiver.query [p.42] to return a list of receivers that are configured on the device.

Feedback messages

Assuming that the specified transport protocol is HTTP or HTTPS, feedback messages follow the format used by the device 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.

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 MAC address of the device.
- **`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 gateway can publish these feedback events:

Event	Description
restart	Sent when the device starts up.
configureAck	Sent when an application successfully configures or reconfigures a feedback receiver.
connectionFinished	Sent when a call has finished on the device. This event prompts the application to retrieve recent call history details for the device. An example of its intended use is: 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 about the call that has finished.
isdnSettingsModified	Sent when the ISDN configuration settings for the device are modified.
dialPlanModified	Sent when the dial plan settings for the device are modified.

system.xml information

You can derive some information about the device from its system.xml file, which you can download via HTTP from the device root (for example, **http://ISDNGW/system.xml**). Information available in the system.xml file includes the manufacturer, model type, and serial number of the device.

Example system.xml

```
<?xml version="1.0"?>
  <system>
    <manufacturer>Cisco/manufacturer>
    <model>ISDN GW 3241</model>
    <serial>SM000C00</serial>
    <chassisSerial>XX7000E9</chassisSerial>
    <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>
```

Fields in the system.xml file

Field	Description
manufacturer	Device manufacturer.
model	Device model type.
serial	Device serial number.
softwareVersion	Software version that is currently running.
buildVersion	Build version of the currently running software.
hostName	System host name.
uptimeSeconds	Number of seconds since boot.

Fault codes

In common with certain other Cisco TelePresence applications, the Cisco TelePresence ISDN Gateway returns a fault code when a fault occurs during processing of an XML-RPC request.

The individual call descriptions in this guide give some indication of which faults may occur. The following table describes all possible fault codes that are used within this specification and their most common interpretations. 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**. The 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.

Fault code	Description
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**. An ISDN or serial port is given as a parameter but does not exist on an ISDN or serial gateway device.
37	**Dial plan full**. An attempt was made to add a rule to a dial plan which already contains the maximum permitted number of rules (200).
38	**Dial plan rule not found**. An attempt was made to modify or delete a non-existent rule.
39	**Invalid dial plan version**. A request to modify the dial plan provided an incorrect (probably outdated) revision number.
40	**Invalid regex groups**. A pattern was entered with too many groups, or with more substitutions than groups.
47	**Dial plan invalid ordering**. The new ordering provided for the dial plan is invalid.
59	**Conflicting parameter values**. Contradictory values were provided for parameters (such as specifying 30-channel maximum call rate with interface type T1).
101	**Missing parameter**. 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**. A parameter was successfully parsed and 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.

Related information

All documentation for the latest versions of the Cisco TelePresence products covered in this guide can be found on Cisco.com. Other documents referred to in this guide can be found at:

Title	Reference	Link
RFC 2616: Hypertext Transfer Protocol - HTTP/1.1	RFC 2616	http://www.faqs.org/rfcs
XML-RPC specification		www.xmlrpc.com