Cisco Unified Communications Gateway Services API Guide

2020-09-25

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide.
Addresses, phone numbers, and fax numbers
are listed on the Cisco website at
www.cisco.com/go/offices.

CONTENTS

Cisco Unified Communications Gateway Services API

This chapter describes the Cisco Unified Communications Gateway Services Application Programming Interface (CUCGSAPI). The CUCGSAPI enables the development of advanced Cisco Unified Communication applications and services on the platforms running Cisco IOS and Cisco IOS XE software by providing an interface to the Cisco Unified Communications Gateway Services.

CUCGSAPI provides the developer with access to the following unified communications gateway services:

- Extended Call Control Service.
- Extended Serviceability Service.
- Extended Call Detail Record (CDR) Service.
- Extended Media Forking.

Feature Information for Cisco Unified Communications Gateway Services

Feature Name	Release	Feature Information
Cisco Unified Communications Gateway Services	Cisco IOS 15.2(2)T Cisco IOS XE 3.10	The Cisco Unified Communications Gateway Services API provides a unified web services interface for the different services in IOS gateway thereby facilitating rapid service development at application servers and managed application service providers.
Cisco Unified Communications Gateway Services API support for Secure RTP Forking	Cisco IOS 15.4(3)M Cisco IOS XE 3.13S	This feature provides support for Extended Media Forking (XMF) provider to monitor calls and trigger media forking on RTP and SRTP calls.
HTTPS support for Cisco Unified Communications Gateway Services	Cisco IOS XE Everest 16.6.1	This feature allows Cisco Unified Communications Gateway Services to establish a secure connection using HTTPS protocol with the application.

Overview

CUCGSAPI allows you to develop an application that interacts with the Cisco Unified Communications Gateway Services on voice gateways. The application accesses the Cisco Unified Communications Gateway Services via SOAP messages.

You can configure secure and nonsecure modes of connectivity between Cisco Unified Communications Gateway Services and application. Nonsecure mode uses HTTP protocol and secure mode uses HTTPS protocol.

Figure 1-1 illustrates the Cisco Unified Communications Gateway Services interface in nonsecure mode. In nonsecure mode, Cisco supports the extended call control (XCC) provider, extended call detail record (XCDR) provider, extended serviceability (XSVC) provider, and extended media forking (XMF).

Figure 1-1 Cisco Unified Communications Gateway Services Interface in nonsecure mode

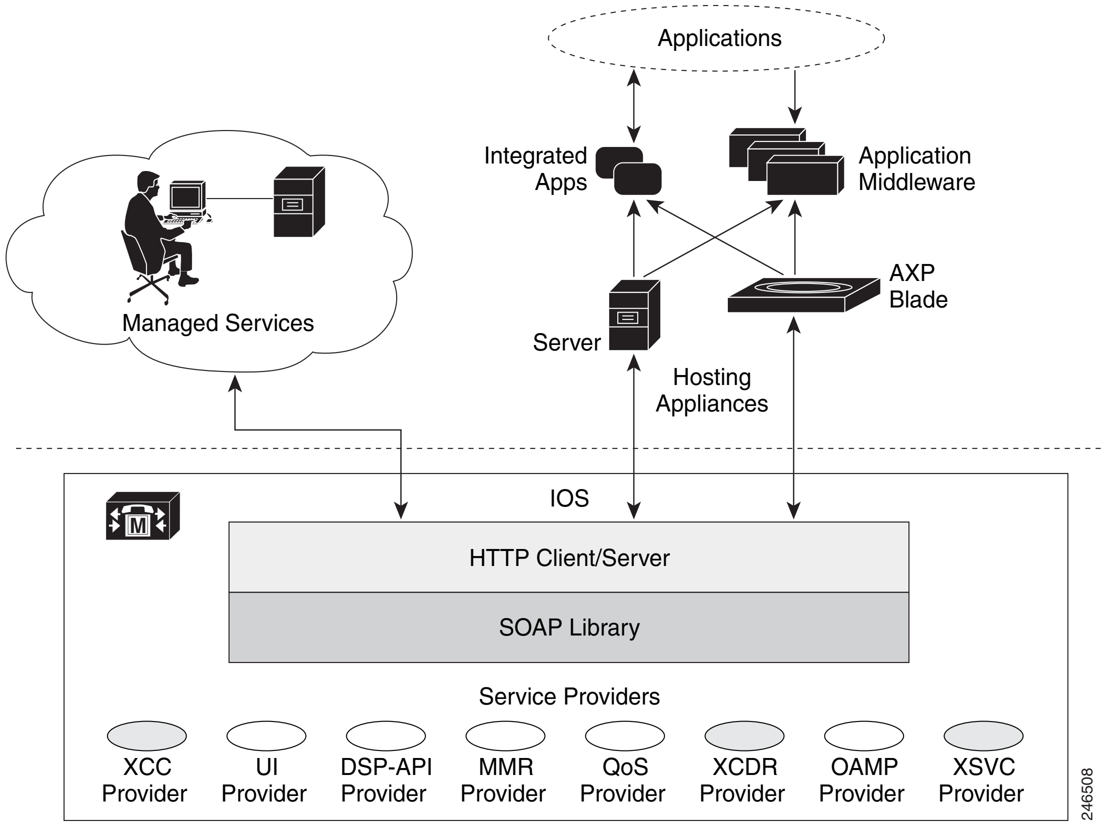
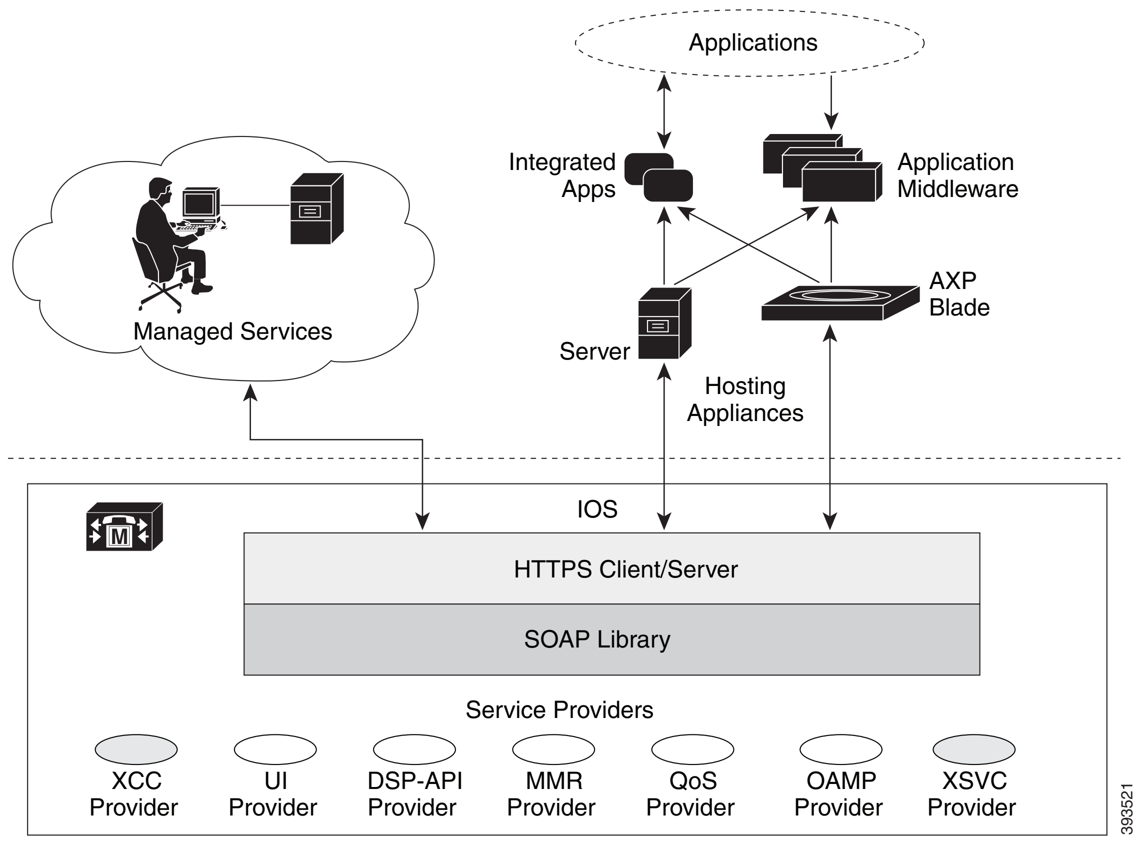


Figure 1-2 illustrates the Cisco Unified Communications Gateway Services interface in secure mode. In secure mode, Cisco supports the extended call control (XCC) provider, and extended serviceability (XSVC) provider.

Figure 1-2 Cisco Unified Communications Gateway Services Interface in secure mode



Cisco Unified Communications Gateway Services

Web service is a standards-based framework that allow applications operating on different platforms to interact over the Internet. Cisco Unified Communications Gateway Services, like web services, are platform independent and language neutral. With CUCGSAPI, you can develop your application in the language and operating system of your choice and communicate directly with the Cisco Unified Communications Gateway Services running on the voice gateway.

The Cisco Unified Communications Gateway Services API supports the following standards and protocol:

- XML 1.0
- Web Services Description Language (WSDL) 1.1
- SOAP, version 1.2
- HTTP, version 1.1
- HTTPS, version 1.1
- TLS, version 1.1, 1.2

Behavioral Differences Between Nonsecure Mode and Secure Mode

Table 1-1 lists the differences in behavior between nonsecure and secure mode.

Table 1-1 Behavioral Differences Between Nonsecure and Secure Mode

Behavior	Nonsecure Mode	Secure Mode
Application Protocol	HTTP	HTTPS
Transport Protocol	TCP	TCP-TLS (1.1, 1.2)
Supported Providers	XCC, XCDR, XSVC, XMF	XCC, XSVC
Listen Port	8090 (fixed)	443 (configurable)
Platforms Supported	Cisco IOS software based platforms and Cisco IOS XE software based platforms	Cisco IOS XE software based platforms
Number of Remote URLs supported for XSVC provider	Up to 8	1

Limitations

- Trunk status monitoring via XSVC is not supported on Cisco IOS XE Software Releases.
- Virtual Routing and Forwarding (VRF) or Multi-VRF is not supported with Web Services API (WSAPI) feature on IOS Voice Gateways.

Providers

The providers on the voice gateway provide services on the voice gateway for remote applications. Cisco Unified Communications Gateway Services API enables applications to interact with the providers and is comprised of the following provider objects:

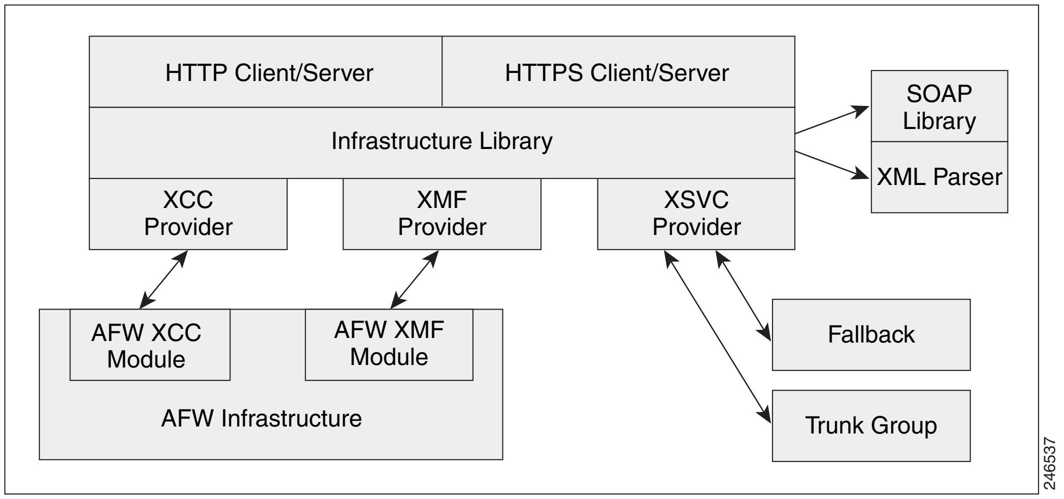
- XCC Provider— Extended Call Control (XCC) provider supports operations that allow an application to perform call control and real-time call monitoring.
- XCDR Provider—Extended Call Detail Record (XCDR) provider supplies CDR information to the application and notifies the application when calls have ended.
- XSVC Provider—Extended Serviceability (XSVC) provider monitors trunk status, and provides real-time link status and configuration change notification to application.
- XMF Provider—Extended Media Forking (XMF) provider monitors calls and trigger media forking on the calls and has the capability to service up to 32 applications.

Each provider has a unique URL identifier and communicates with the application via SOAP messages. The providers can be in one of two states:

- In-service—Provider is active and available for use.
- Shutdown—Provider is disabled and no longer available. The API methods associated with this provider are invalid in this state.

Figure 1-3 illustrates the relationship between the IOS components.

Figure 1-3 Cisco Unified Communications Gateway Services Components



When a provider is configured and enabled on the voice gateway, it performs the following functions:

- Manages the registration process between the application and the provider.
- Sends notification to the application when a provider changes its status.
- Passes incoming messages to the appropriate provider.
- Notifies the provider when there is a message exchange failure.
- Sends probing messages to maintain an active registration session.
- Sends negative probing messages to detect the status of an application. If the number of failed responses exceeds a configured number of negative probing messages, the voice gateway unregisters the application.

WSDL Files

CUCGSAPI uses the WSDL specification to define the services that are available on the voice gateway. These services are represented as providers on the voice gateway.

Table 1-2 lists the namespace for the Cisco Unified Communications Gateway Services

Table 1-2 Cisco Unified Communications Gateway Services Namespace

Service	Location
XCC	http://www.cisco.com/schema/cisco_xcc/v1_0
XCDR	http://www.cisco.com/schema/cisco_xcdr/v1_0
XSVC	http://www.cisco.com/schema/cisco_xsvc/v1_0
XMF	http://www.cisco.com/schema/cisco_xmf/v1_0

Inbound Ports

Table 1-3 lists the URL and inbound location that the application uses to communicate with the server in nonsecure mode.

Table 1-3 Location of the Inbound Port in Nonsecure Mode

Service	Namespace
XCC	http://<access_router>:8090/cisco_xcc [1]
XCDR	http://<access_router>:8090/cisco_xcdr [1]
XSVC	http://<access_router>:8090/cisco_xsvc [1]
XMF	http://<access_router>:8090/cisco_xmf [1]

1. The access_router is the hostname or IP address of the router that with Cisco Unified Communications Gateway Services.

Table 1-4 lists the URL and inbound location that applications uses to communicate with the server in secure mode.

Table 1-4 Location of the Inbound Port in Secure Mode

Service	Namespace
XCC	https://<access_router>:443/cisco_xcc [2]
XSVC	https://<access_router>:443/cisco_xsvc [2]

2. HTTPS connection uses inbound port 443 by default. Execute **ip http secure-port** *port* command if you want to change the default port

Registering an Application

Before an application can register with the voice gateway, you must first configure the application's service URL on the voice gateway. The URL is used to authenticate messages from the application. When the voice gateway first boots up, the provider sends status messages to the applications that are in its configuration. The voice gateway sends status messages when the provider changes its status.

The application initiates registration by sending a registration message to the appropriate provider. The provider generates a unique registration ID and sends it back to the application. The unique registration ID identifies the registered session and is used in all messages that are sent during the registered session.

States of a Registered Session

The state of the registered session and the status of the messages that are sent between the provider and application have one of the following value:

- Steady State—This state is the normal state of the registered session. Messages and acknowledgements are exchanged regularly in this state.
- Keepalive State—When the provider does not have messages to send, the voice gateway sends keepalive probing messages to the registered application This keeps the connection between the application and the provider active. The messages that are sent in this state contain information on the health and connectivity status of the provider.
- Negative Probe State—When the number of failed responses exceeds the maximum number of failed responses, the registered session enters the negative probe state. In the negative probe state, the voice gateway sends negative-probing messages in an attempt to reestablish the steady state or the

keepalive state with the application. The only message sent in a negative probe state is a negative probe message. The registered session returns to a steady state or keepalive state upon receipt of a successful response to a negative probe message, and regular messages resume.

- Unregistered State—The session is unregistered and no messages are exchanged between the provider and the application. The session enters an unregistered state under the following conditions:
  - When the application unregisters with the provider.
  - When an application fails to respond to probing messages.
  - When the administrator shuts down the provider service on the voice gateway.

XCC Provider

The XCC provider gives an application the capability to control all the legs of a standard call. With the XCC provider, the application can perform auxiliary call control and can control some network elements.

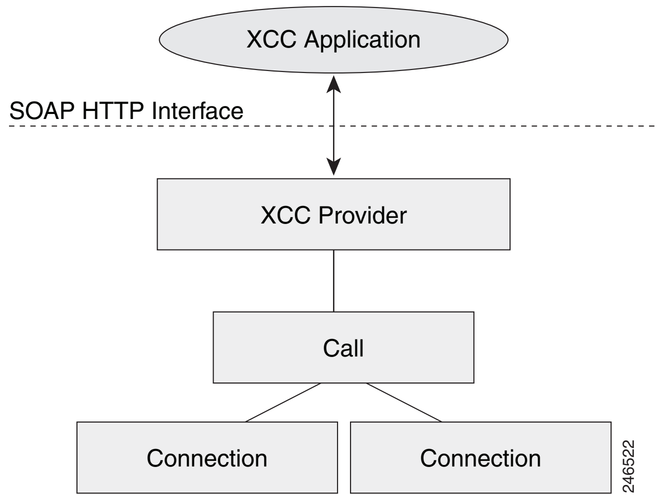
Characteristics of the XCC Provider

The XCC provider has the following characteristics:

- The XCC provider allows the application to maintain stateful control on a call over the entire life cycle of the call.
- The XCC provider allows the application to subscribe and receive mid-call event notification. The application can change event subscription over the life of the call.
- The XCC provider allows services to be invoked on a network triggered event. The provider reports on notifications from a direct application request.
- The XCC provider follows a generic call model in which the underlying communication protocol and architecture is hidden from the developer. XCC provider uses a high-level call control model for maintaining and managing the state of a call session.

Figure 1-4 illustrates the XCC call control abstraction.

Figure 1-4 XCC Call Control



XCC Provider API

When an application registers with the XCC provider, the application configures event filter parameters that the application is interested in monitoring, and the XCC provider installs a connection listener to monitor the calls. XCC notifies the application when a call or connection event matches the event filters that were configured. When the application updates event filter parameters, the updates only apply to new calls, not existing calls.

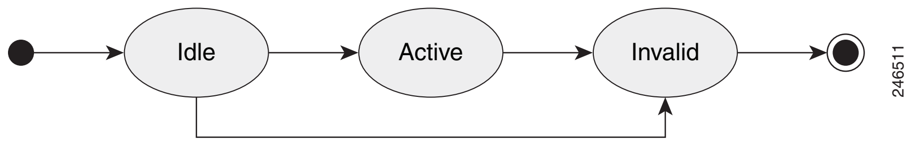
The XCC provider API is described in XCC Provider Operations.

The XCC call APIs describe the endpoints and trunks that are associated with a call. The APIs in XCC call API, and the associated XCC connection describes the control and media flow in a call. The provider notifies the application when there is a change to the state of a call and sends update information on the call, address, and connections.

A call abstraction is represented in Figure 1-5 on the voice gateway in one of the following three states:

- IDLE—Initial state of all calls. A call in an idle state has zero connections.
- ACTIVE—Call with ongoing activity. A call in an active state has one or more associated connections.
- INVALID—Final state of all calls. Calls that lose all their connections are moved into this state. A call in the invalid state has zero connections.

Figure 1-5 Call Abstraction Model



XCC Call Media Set Attributes

External applications can enable the voice gateway to detect changes to the call media set attributes on a call and have the voice gateway send a notify event message. Table 1-5 lists the call media set attributes that the gateway can detect.

Table 1-5 Call Media Set Attributes

Call Media Set Attributes	Description
Call Mode Change	Enables the voice gateway to detect when a call changes between the following call modes: • Voice Call • Fax Call • Video Call • Modem Call • Data Call **Note** ISDN calls with an unrestricted bearer capability value are reported as data calls.
DTMF	Enables the voice gateway to detect a DTMF digit in the media stream or a DTMF relay. **Note** The notify event message includes the timestamp if the DTMF event is detected in IOS. **Note** For notify event messages, the application should use the voice gateway as the NTP[1] server for synchronizing clocks.
Media Activity	Enables the voice gateway to detect when the media activity state changes from "Active" to "Inactive" or vice versa.
Tone	Enables the voice gateway to detect the following specified tones: • Busy Tone • Dial Tone • Ringback Tone • Out-of-Service Tone • Second Dial Tone **Note** Tone detection is not supported for a FXO voice port if the supervisory tone detection feature is enabled.
Media Forking	Enables media forking on a connected call to target a RTP address. For more information on media forking, see the "XCC Call Media Forking" section on page 1-9.

1. NTP = network time protocol.

XCC Call Media Forking

Note XCC Call Media Forking is not supported in secure mode.

External applications can request media forking for a call. When the application requests media forking, it must provide the XCC provider with two unique remote RTP ports (nearEndAddr and farEndAddr). The XCC provider identifies the incoming connection of a call, forks both the transmit (TX) and receive (RX) packets, and sends the packets to the targeted RTP ports. The XCC provider uses the nearEndAddr element for the forked TX media stream and the farEndAddr XCC element to record the RX media stream. The two forked media streams are sent from the voice gateway in a "SEND ONLY" direction.

Media forking has the following limitations:

- Supports only voice media stream.
- Supports only IPv4 RTP forked media stream.
- Media mixing on forked media streams is not supported.
- Media negotiation is not supported on the forked media streams. In other words, the codec of the forked media stream cannot be changed. If the targeted media server supports a dynamic codec format in the forked media stream, you must configure a supported codec, such as G.711, in the voice gateway.
- Media renegotiation is not supported.
- Media forking ends when the connection is disconnected.
- Supplementary services are not supported.
- Only one media forking request per session is supported. The XCC provider rejects additional media forking request from the application.
- TDM call legs cannot be forked. If CallID is used, it always anchors the fork to the inbound call leg, which must be IP based.

The XCC provider updates the application on the status of the media forking by including one of the following states in the NotifyXccCallData message.

- FORK_FAILED—Setup for media forking failed. Forked RTP connections cannot be established with the targeted RTP addresses.
- FORK_STARTED—Media forking was successful. Both the TX and RX forked RTP connections are established and connected to the targeted RTP addresses.
- FORK_DONE—Media forking has completed. Both the TX and RX forked RTP connections are released.

XCC Connection

The XCC connection describes the relationship in a XCC call and the endpoint or trunk in the call. Figure 1-6 illustrates the connection states.

Figure 1-6 Connection States

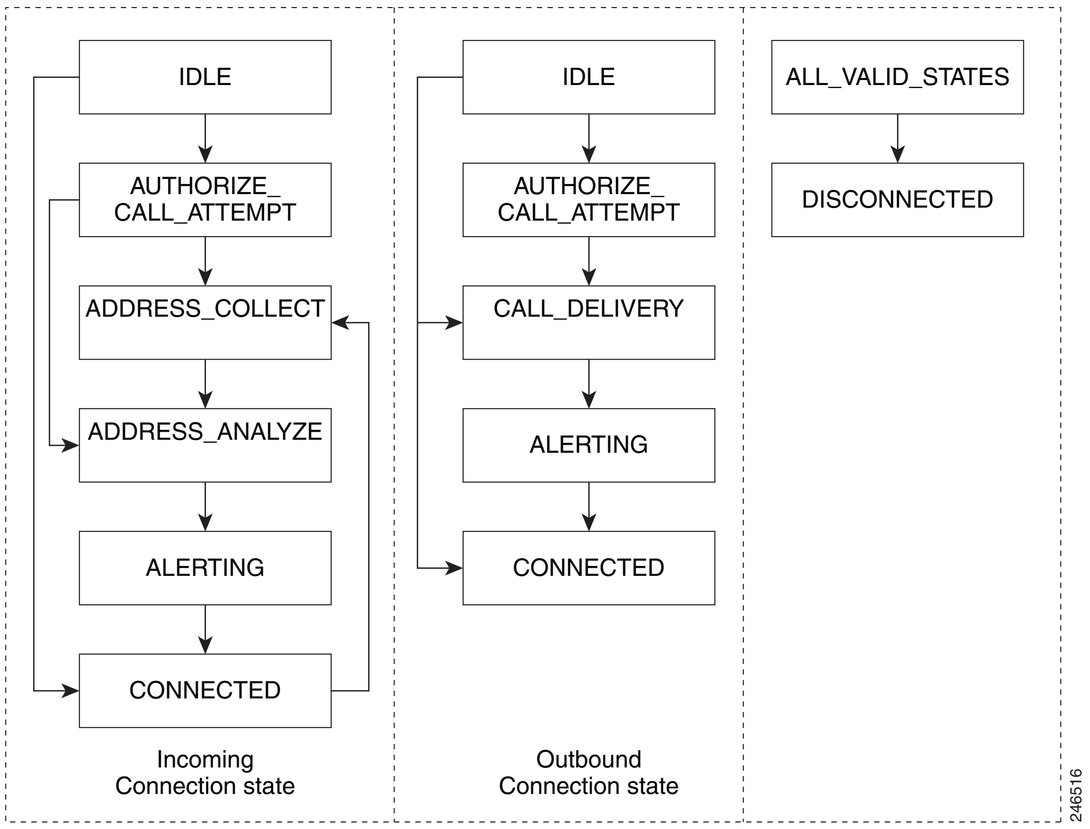


Table 1-6 describes the connection states and the activity and exchanges that can occur between the voice gateway and application when the application sets up event notifications for a particular connection state.

Table 1-6 Connection States

Connection States	Description	Activity and Messages sent between the Voice Gateway and Application
IDLE	Initial state of all new connections. In this state, the connection is not an active part of the call, but references to the call and address are valid.	**Voice Gateway** The voice gateway sends a NotifyXccConnectionData(CREATED) message for inbound calls. No messages are sent for outbound calls.
AUTHORIZE_CALL_ATTEMPT	Originating endpoint is waiting for authorization.	**Voice Gateway** The voice gateway places the call in a suspended state, sends a SolicitXccConnectionAuthorize() message, and waits for a response from the application. **Application** The application sends the ResponseXccConnectionAuthorize() message directing the gateway to either continue processing or release the call.
ADDRESS_COLLECT	Gateway is collecting information from the originating party.	No messages are sent.
ADDRESS_ANALYZE	Gateway has finished collecting the originating party information and is analyzing and translating the information according to a dial plan.	**Voice Gateway** The voice gateway places the call in a suspended state, sends a SolicitXccConnectionAddressAnalyze() message, and waits for a response from the application. **Application** The application sends either the call route back to the gateway or delegates the voice gateway to make the route selection in the ResponseXccConnectionAddressAnalyze () message.
CALL_DELIVERY	On an outbound call, the voice gateway selects the route and sends a request that a call be setup at the specified called endpoint.	No messages are sent for inbound calls. **Voice Gateway** The voice gateway sends a NotifyXccConnectionData(CREATED) and a NotifyXccConnectionData(CALL) DELIVERY) message for outbound calls.
ALERTING	Endpoint is being notified of the incoming call.	**Voice Gateway** The voice gateway sends a NotifyXccConnectionData(ALERTING) message.

Table 1-6 Connection States (continued)

Connection States	Description	Activity and Messages sent between the Voice Gateway and Application
CONNECTED	Connection and address for the call active.	**Voice Gateway** The voice gateway sends a NotifyXccConnectionData (CONNECTED) message.
DISCONNECTED	Connection is no longer active.	**Voice Gateway** The voice gateway sends a NotifyXccConnectionData(DISCONNECTED) message.

XSVC Provider

The extended serviceability provider (XSVC provider) monitors the health of the trunk and provides the application with real-time trunk status.

The XSVC provider can monitor both traditional public switched telephone network (PSTN) trunks and VoIP trunks. You must configure the XSVC provider and install a route listener for XSVC on the interested trunk group to begin monitoring the trunk status. The route listener communicates with the trunk group resource manager to obtain information on the trunks, including alarm information for T1/E1 trunks.

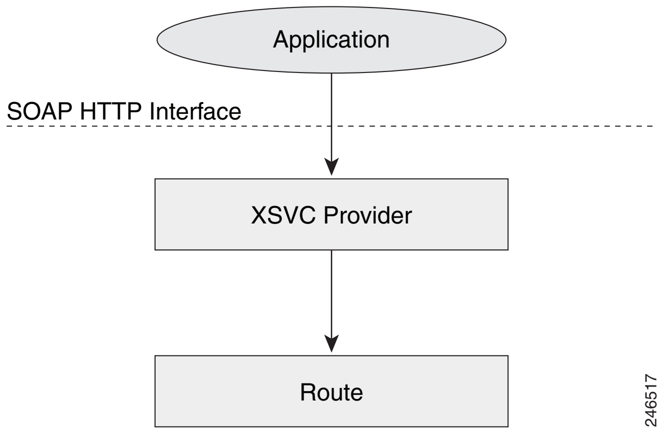
For PSTN trunks, the trunk group is a logical grouping of interfaces with the same signaling characteristics, such as DS1, FXO, or PRI interfaces. The trunk group can have more than one PRI interface and can also support FXO, but you cannot mix FXO and T1/E1 interfaces. The trunk group resource manager supports the logical configuration of trunk groups.

For VoIP trunks, the trunk manager monitors a VoIP trunks by using Internet Control Message Protocol (ICMP) pings. The trunk manager supports up to 1000 trunks.

When the application registers with the XSVC provider, the application obtains a handler that the application uses to receive snapshot information on all the routes or specific routes. The XSVC provider can support up to 8 different applications, with each application able to monitor a particular group of trunks.

Figure 1-7 illustrates the relationship between the application, XSVC route, and XSVC provider.

Figure 1-7 XSVC Provider



Characteristics of the XSVC Provider

The XSVC provider has the following characteristics:

- When the XSVC provider cannot reach the remote application, the XSVC provider discards event information messages.
- The application must register with the XSVC provider or use a snapshot to obtain the most updated trunk information.
- During the registration, the application can configure event filters for a registered session. The event filters only applies for that registered session.
- The XSVC provider reports on the current status of the trunk. The XSVC provider does not report on changes to a trunk configuration until the change has taken effect.

XSVC Provider API

When the application registers with the XSVC provider, a route listener is installed on the trunk interfaces. If filters are not specified in the registration message, the XSVC provider does not filter out any events. For the application to receive the most current trunk configuration, we recommend that you do not filter out the ROUTE_CONF_UPDATED event.

The XSVC provider API is described in Xsvc Provider Operations.

XSVC Route

With the route snapshot API, the application can request and receive a summary from the voice gateway on all the routes that are currently being monitored in a compact format. The application can also set up a filter to listen to specific routes. The application can also request that the XSVC provider send detail information for a specific route. For T1/E1 trunks, the XSVC provider sends additional information, such as channels, total available channels, alarm, and error statistics.

Alarm Definition

Table 1-7 describes the alarm definition that can be found in XSVC route messages.

Table 1-7 Alarm Definition

Alarm	Definition
NoAlarm	No alarm present
RcvFarEndLOF	Far end LOF[1] indication (a.k.a. Yellow Alarm)
XmtFarEndLOF	Near end sending LOF indication
RcvAIS	Far end sending AIS[2]
XmtAIS	Near end sending AIS
LossOfFrame	Near end LOF (a.k.a. Red Alarm)
LossOfSignal	Near end loss of signal
LoopbackState	Near end has a loop back
T16AIS	E1 TS16 AIS
RcvFarEndLOMF	Far end is sending TS16 LOMF[3]
RcvFarEndLOMF	Near end is sending TS16 LOMF
RcvTestCode	Near end detects a test code
OtherFailure	Line status that is not defined here
UnavailSigState	Near end is in an unavailable signal state
NetEquipOOS	Carrier equipment is out of service
RcvPayloadAIS	DS2 payload AIS
Ds2PerfThreshold	DS2 performance threshold

1. LOF = loss of frame.
2. AIS = alarm indication signal.
3. LOMF = loss of multiframe.

Statistics Definition

Table 1-7 defines the statistics that are collected and can be found in XSVC route messages.

Table 1-8 Statistics Definition

Statistics	Definition
LCV	Line Coding Violation Error Event
PCV	Path Coding Violation Error Event
CSS	Controlled Slip Seconds
SEFS	Severely Errored Frame Seconds
LES	Line Errored Seconds
DM	Degraded Minutes
ES	Errored Seconds

Table 1-8 Statistics Definition (continued)

Statistics	Definition
BES	Bursty Errored Seconds
SES	Severely Errored Seconds
UAS	Unavailable Seconds

XCDR Provider

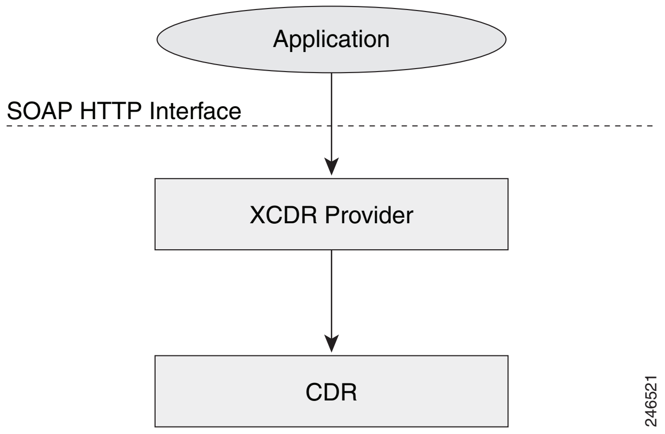
The XCDR provider sends information on a call detail record (CDR) to the registered application when a call ends. The CDR contains statistics on the call and calling party and called party information in a CSV format. The XCDR provider can support up to eight remote application.

When the application registers with the XCDR provider, it obtains a handler that the application can use to receive CDR records. The application can choose to receive either the compact or detailed CDR format.

Note By default, the XCDR provider sends out the CDR record in a compact format to save bandwidth.

Figure 1-8 illustrates the relationship between the application, CDR, and XCDR provider.

Figure 1-8 XCDR



XCDR Provider API

The XCDR provider API is described in Xcdr Provider Operations.

XCDR CDR is responsible for collecting CDR information and generating events that are sent to the application. The application can specify whether it wants the CDR record in compact or detailed format by using the RequestXcdrSetAttribute message.

Call Detail Record

For detail information on the name and order of the call detail record fields, see *CDR Accounting for Cisco IOS Voice Gateways*.

XMF Provider

XMF provider gives an application the capability to monitor calls and trigger media forking on the calls and has the capability to service up to 32 applications. The XMF provider can invoke a call-based or a connection-based media forking using the Unified Communications (UC) API. After the media forking is invoked, it can preserve the media forking initiated by the web application if the WAN connection to the application is lost. The XMF provider also provides the recording tone to the parties involved in the call.

The XMF provider API is described in Xmf Provider Operations.

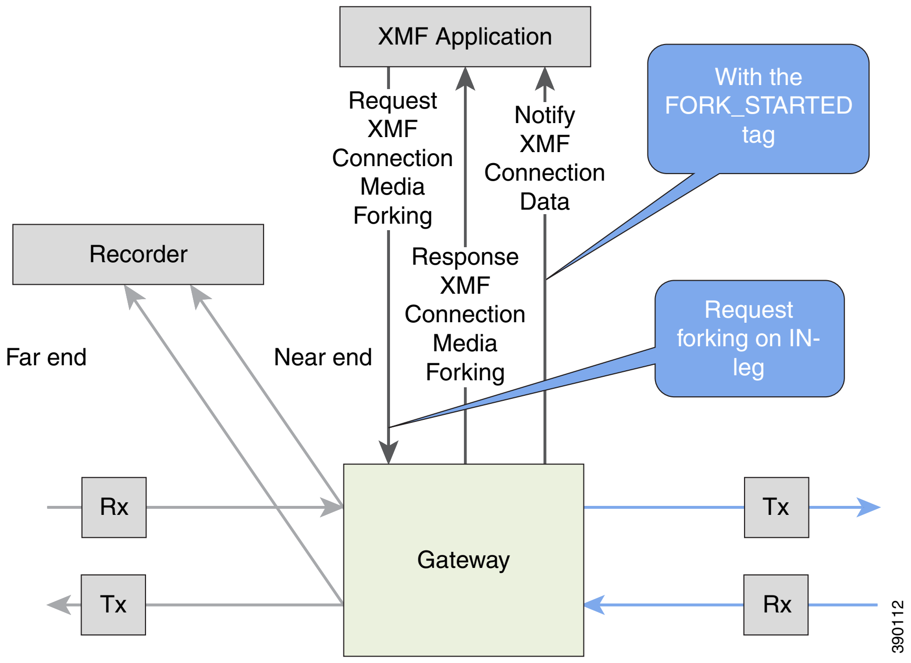
XMF Call-Based Media Forking

In call-based media forking of the gateway, the stream from the calling party is termed as near-end stream and the stream from the called party is termed as far-end stream. The XMF provider actively handles single media forking request per session. Any new media forking request from the external application will override or stop the current forking instance and would start a new forking instance (to the appropriate target IP address or ports). After the media forking request is accepted, the XMF provider returns a response message and starts to fork media streams of a connection to the target forked streams. A NotifyXmfCallData message will be notified to the application for the updated media forking status, that is, FORK-FAILED, FORK_STARTED, or FORK_DONE.

XMF Connection-Based Media Forking

In connection-based media forking of the gateway, the incoming stream to the connection is termed as near-end stream and the outgoing stream of the connection is termed as far-end stream. The XMF provider actively handles single media forking request per session. Any new media forking request from the external application will override or stop the current forking instance and would start a new forking instance (to the appropriate target IP address or ports). After the media forking request is accepted, the XMF provider returns a response message and starts to fork media streams of a connection to the target forked streams.

Figure 1-9 XMF Connection Based Media Forking


XMF Application
Request XMF Connection Media Forking
Response XMF Connection Media Forking
Notify XMF Connection Data
With the FORK_STARTED tag
Request forking on IN-leg
Recorder
Far end
Near end
Rx
Tx
Gateway
Tx
Rx
390112

A NotifyXmfConnectionData message will be notified to the application for the updated media forking status:

- FORK_FAILED—Media forking is setup failure. No forked RTP connections can be established to target RTP addresses.
- FORK_STARTED—Media forking is set up successfully. Both Tx (transmit) and Rx (receive) forked RTP connections are established and connected to target (farEnd and nearEnd) RTP addresses.
- FORK_DONE—Media forking is completed. Both Tx and Rx forked RTP connections are released.

XMF Connection

The XMF connection describes the relationship between an XMF call and the endpoint (or trunk) involved in the call. A connection abstraction maintained in the gateway has the following connection states:

Table 1-9 XMF Connection States

Connection Status	Description
IDLE	Initial state for all new connections. Such connections are not actively part of a telephone call, yet their references to the Call and Address objects are valid. Connections typically do not stay in the IDLE state for long and quickly transition to other states. The application may choose to be notified at this state using the event filters and if done, call/connection at the gateway provider will use the NotifyXmfConnectionData(CREATED) message to notify the application listener that a new connection is created.
ADDRESS_COLLECT	In this state the initial information package is collected from the originating party and is examined according to the "dialing plan" to determine the end of collection of addressing information. In this state, the call in the gateway collects digits from the endpoint. No notification is provided.
CALL_DELIVERY	On the originating side, this state involves selecting of the route as well as sending an indication of the desire to set up a call to the specified called party. On the terminating side, this state involves checking the busy/idle status of the terminating access and also informing the terminating message of an incoming call. The application may choose to be notified at this state using the event filters and if done, the call or connection at the gateway provider will use the NotifyXmfConnectionData (CALL_DELIVERY) message to notify the application listener.
ALERTING	This state implies that the Address is being notified of an incoming call. The application may choose to be notified at this state using the event filters and if done, the call or connection at the gateway provider will use the NotifyXmfConnectionData (ALERTING) message to notify the application listener.

Table 1-9 XMF Connection States

Connection Status	Description
CONNECTED	This state implies that a connection and its Address is actively part of a telephone call. In common terms, two parties talking to one another are represented by two connections in the CONNECTED state. The application may choose to be notified at this state using the event filters and if done, the call or connection at the gateway provider will use the NotifyXmfConnectionData (CONNECTED) message to notify the application listener.
DISCONNECTED	This state implies it is no longer part of the telephone call. A Connection in this state is interpreted as once previously belonging to this telephone call. The application may choose to be notified at this state using the event filters and if done, the call or connection at the gateway provider will use the NotifyXmfConnectionData (DISCONNECTED) message to notify the application listener.

Media Forking for SRTP Calls

SRTP forking is supported in XMF and XCC application service providers and the supported APIs are RequestCallMediaForking, RequestCallMediaSetAttributes, and RequestConnectionMediaForking.

SRTP forking is supported for SRTP-to-SRTP, SRTP-to-RTP, and RTP-to-SRTP calls.

- For SRTP-to-SRTP calls, media forking on either leg would result in SRTP streams being forked.
- For SRTP fallback calls, after the initial offer, voice gateway will fall back to RTP. Media forking either call legs would result in RTP streams being forked.
- For SRTP-to-RTP interworking calls, a digital signal processor (DSP) is required and involves transcoding. In this case, one leg would be SRTP and the other leg RTP.

SRTP Crypto keys are notified over the API.

Supports automatic stopping of media forking when stream changes from SRTP or to SRTP.

- The optional mediaForkingReason tag in XMF or XCC Notify messages indicates that the forking has been stopped internally.
- mediaForkingReason tag is only present when the connection changes state, such as mid-call re-INVITE. SRTP stream can change to RTP or SRTP stream can change keys mid-call.
- mediaForkingReason tag is always accompanied by FORK_DONE.

Crypto Tag

For SRTP forking, the optional Crypto tag in NotifyXmfConnectionData or NotifyXmfCallData message indicates the context of an actively forked SRTP connection.

Note The Crypto tag is only present in the notification message where FORK_STARTED tag is present.

The optional Crypto tag specifies the following:

- The Crypto suite used for encryption and authentication algorithm.
- The base64 encoded mastery key and salt used for encryption.

Crypto suite can be one of the two suites supported in IOS:

- AES_CM_128_HMAC_SHA1_32
- AES_CM_128_HMAC_SHA1_80

The following is a sample SDP data sent in an SRTP call:

Table 1-10 Sample SDP Data Sent in an SRTP Call

Original SIP SDP Offer	SIP SDP Crypto Answer
v=0	v=0
o=CiscoSystemsSIP-GW-UserAgent 7826 3751 IN IP4 172.18.193.98	o=CiscoSystemsSIP-GW-UserAgent 7826 3751 IN IP4 172.18.193.98
s=SIP Call	s=SIP Call
c=IN IP4 172.18.193.98	c=IN IP4 172.18.193.98
t= 0 0	t=0 0
m=audio 51372 RTP/SAVP 0	m=audio 49170 RTP/SAVP 0
a=rtpmap:0 PCMU/8000	**a=crypto:1 AES_CM_128_HMAW_SHA1_32**
a=crypto:1 AES_CM_128_HMAC_SHA1_32 **inline:d0RmdmcmVCspEc3QGZiNWpVLFJhQX1cfHAwJSoj**	**inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj**

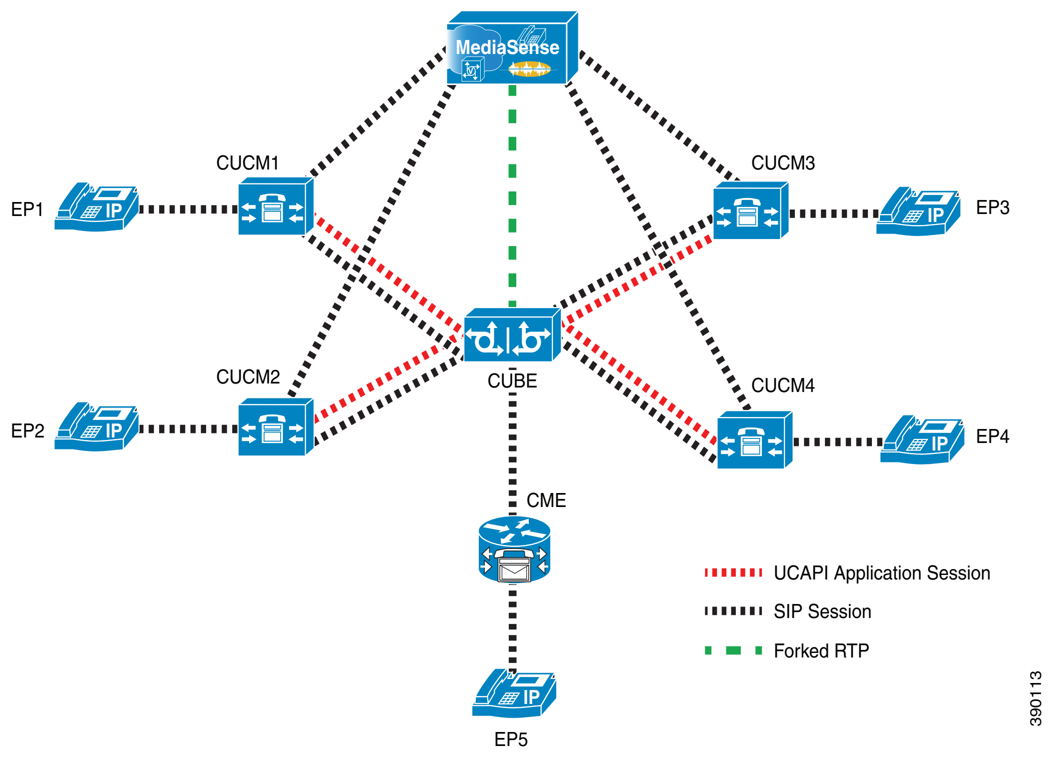
Note The application is notified of the content in Crypto and inline SDP lines.

Multiple XMF Applications Recording Tone

Multiple XMF allows multiple (maximum 32) web applications to register with the XMF provider as separate XMF applications and provide redundancy for the voice calls recording. Recording tone provides recording tone capability to the recording sessions. Recording tone is supported for IP to IP, IP to TDM, and TDM to TDM trunks.

An example topology is as shown below where 4 CUCM applications are deployed. CUCM triggers media forking request to voice gateway. Recording tone is played to the parties involved in the call based on the recordTone parameter set in the media forking request.

Figure 1-10 Multiple XMF Applications and Recording Tone



Media forking can be invoked using any of the following APIs:

- RequestXmfConnectionMediaForking
- RequestXmfCallMediaForking
- RequestXmfCallMediaSetAttributes

The “recordTone” parameter can be enabled in any of the above requests and recording tone will be played for the parties involved in the call. The “recordTone” parameter in the API request can have the following values:

- COUNTRY_US
- COUNTRY_AUSTRALIA
- COUNTRY_GERMANY
- COUNTRY_RUSSIA
- COUNTRY_SPAIN
- COUNTRY_SWITZERLAND

There is no difference in the recording tone beep when any country value is chosen. Recording tone beep is played at an interval of every 15 seconds. Digital signal processors and other resources are not utilized for playing recording tone even for transcoded calls. No specific configuration is required to enable or disable recording tone. By default, no recording tone is enabled.

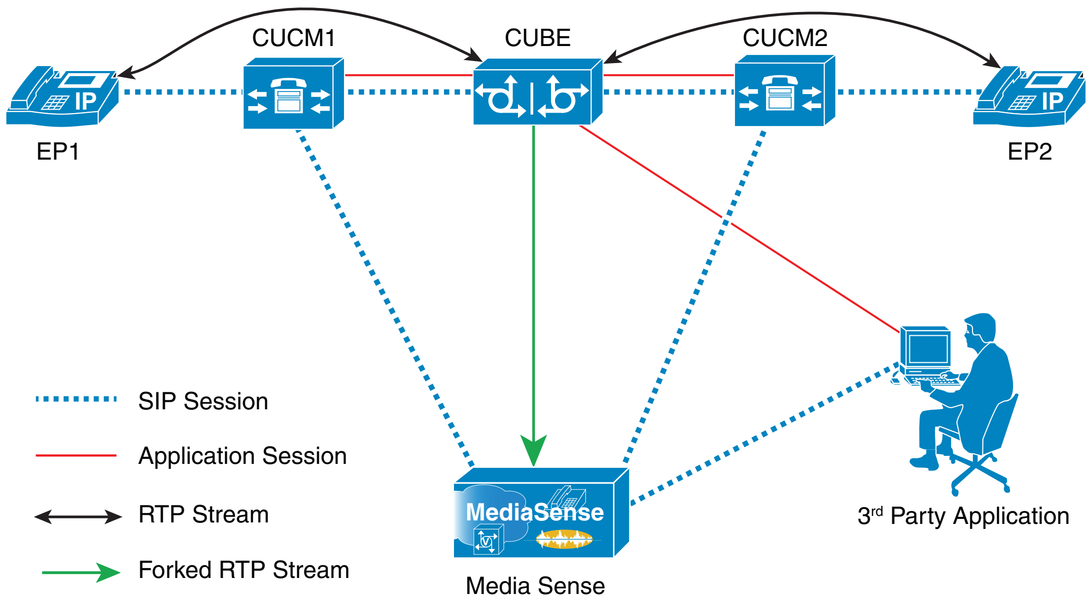
If “recordTone” parameter is enabled only on the farEndAddr, then this tone is played only on the outgoing leg. Likewise, if enabled only on the nearEndAddr, then the tone is played only on the incoming leg. When enabled in both the far and near end, then recording tone is played on both the legs.

The RequestXmfConnectionMediaForking API allows insertion of recording tone on a per connection basis. There could be scenarios where one leg receives two recordTone insertion requests. When a leg receives recordTone insertion request, the nearEnd request always takes precedence over the farEnd request.

Forking Preservation

After media forking is initiated by the web application, the forking can be preserved to continue the recording, even if the WAN connection to the application is lost or if the application is unregistered.

Figure 1-11 Forking Preservation



The “preserve” parameter value can be set to TRUE or FALSE in any of the 3 forking requests (RequestXmfConnectionMediaForking, RequestXmfCallMediaForking, or RequestXmfCallMediaSetAttributes) from the application to voice gateway.

- If the “preserve” parameter received is TRUE, then forking will continue the recording, even if the WAN connection to application is lost or application is unregistered.
- If the “preserve” parameter received is FALSE, then forking will not continue the recording.
- If the “preserve” parameter is not received in the media forking request, then forking will not continue the recording.

CHAPTER 2

Configuring Cisco Unified Communications Gateway Services

Configuring Cisco Unified Communications Gateway Services - Nonsecure Mode

Note If voice gateway is already configured with Cisco Unified Communications Gateway Services in secure mode, remove the secure mode configurations before you proceed with nonsecure mode configuration. Use the **no uc wsapi** command to remove the non-secure mode configuration.

You can configure Cisco Unified Communications Gateway Services in either nonsecure mode or secure mode. When you configure Cisco Unified Communications Gateway Services in nonsecure mode, the command **ip http active-session-modules** *all* is enabled by default, irrespective of whether UC Service APIs provisioned or not. This feature enables all the HTTP applications like UC Gateway Services APIs to register internally for enabling the service. However, if you configure **ip http active-session-modules** *none*, then none of the web applications will register.

To ensure that IOS applications are not enabled by default, configure the following. It explicitly enables web services for specific features of UC Gateway services:

ip http session-module-list [*module_list_name*] [*list_of_modules_to_be_registered*]

ip http active-session-modules [*module_list_name*]

Example

The following is a sample configuration for nonsecure mode. To register only WSAPI services, configure the following:

```
ip http session-module-list wsapi cisco_xmf,cisco_xcc,cisco_xsvc,cisco_xcdr
ip http active-session-modules wsapi
```

To register only XCC services:

```
ip http session-module-list wsapi cisco_xcc
ip http active-session-modules wsapi
```

Prerequisite

- Cisco IOS Release 15.2(2)T or later
- Cisco IOS XE Release 3.10 or later

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip http server**
4. **ip http max-connection** *value*
5. **ip http timeout-policy idle** *seconds* **life** *seconds* **requests** *value*
6. **http client connection persistent**
7. **http client connection idle timeout** *seconds*
8. **uc wsapi**
9. **message-exchange max-failures** *number*
10. **probing max-failures** *number*
11. **probing interval keepalive** *seconds*
12. **probing interval negative** *seconds*
13. **source-address** *ip-address*
14. **end**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`enable`**
	
	Example:
	`Device> enable`
		Enables privileged EXEC mode.
		
		Enter your password if prompted.

Step 2	**`configure terminal`**
	
	Example:
	`Device# configure terminal`
		Enters global configuration mode.

Step 3	**`ip http server`**
	
	Example:
	`Device(config)# ip http server`
		Enables the HTTP server (web server) on the system.

Step 4	**`ip http max-connection`** *`value`*
	
	Example:
	`Device(config)# ip http max-connection 100`
		Sets the maximum number of concurrent connections to the HTTP sever that will be allowed. The default value is 5.

	Command or Action	Purpose

Step 5	**`ip http timeout-policy idle`** *`seconds`* **`life`** *`seconds`* **`requests`** *`value`*
	
	Example:
	`Device(config)# ip http timeout-policy idle 600 life 86400 requests 86400`
		Sets the characteristics that determine how long a connection to the HTTP server should remain open. The characteristics are:
		
		idle—The maximum number of seconds the connection will be kept open if no data is received or response data can not be sent out on the connection. Note that a new value may not take effect on any already existing connections. If the server is too busy or the limit on the life time or the number of requests is reached, the connection may be closed sooner. The default value is 180 seconds (3 minutes).
		
		life—The maximum number of seconds the connection will be kept open, from the time the connection is established. Note that the new value may not take effect on any already existing connections. If the server is too busy or the limit on the idle time or the number of requests is reached, it may close the connection sooner. Also, since the server will not close the connection while actively processing a request, the connection may remain open longer than the specified life time if processing is occurring when the life maximum is reached. In this case, the connection will be closed when processing finishes. The default value is 180 seconds (3 minutes). The maximum value is 86400 seconds (24 hours).
		
		requests—The maximum limit on the number of requests processed on a persistent connection before it is closed. Note that the new value may not take effect on any already existing connections. If the server is too busy or the limit on the idle time or the life time is reached, the connection may be closed before the maximum number of requests are processed. The default value is 1. The maximum value is 86400.

Step 6	**`http client connection persistent`**
	
	Example:
	`Device(config)# http client connection persistent`
		Enables HTTP persistent connections.
		
		Note When this command is configured, multiple files are loaded using the same connection. Executing this command determines whether the HTTP client requests a keepalive or closed connection from the server. The HTTP server is responsible for granting or denying the keepalive connection request from the client.

Step 7	**`http client connection idle timeout`** *`seconds`*
	
	Example:
	`Device(config)# http client connection idle timeout 600`
		Sets the number of seconds that the client waits in the idle state until it closes the connection.

	Command or Action	Purpose
Step 8	**uc wsapi** **Example:** `Device(config)# uc wsapi`	Enters Cisco Unified Communications Gateway Services configuration mode.
Step 9	**message-exchange max-failures** *number* **Example:** `Device(config-uc-wsapi)# message-exchange max failures 2`	Configures the maximum number of failed message exchanges between the application and the provider before the provider stops sending messages to the application. Range is 1 to 3. Default is 1.
Step 10	**probing max-failures** *number* **Example:** `Device(config-uc-wsapi)# probing max-failures 5`	Configures the maximum number of failed probing messages before the voice gateway unregisters the application. Range is 1 to 5. Default is 3.
Step 11	**probing interval keepalive** *seconds* **Example:** `Device(config-uc-wsapi)# probing interval keepalive 180`	Configures the interval between probing messages, in seconds. Default is 120 seconds.
Step 12	**probing interval negative** *seconds* **Example:** `Device(config-uc-wsapi)# probing interval negative 10`	Configures the interval between negative probing messages, in seconds.
Step 13	**source-address** *ip-address* **Example:** `Device(config-uc-wsapi)# source-address 10.25.12.13`	Configures the IP address (hostname) as the source IP address for the Cisco Unified Communications Gateway Services. **Note** The source IP address is used by the provider in the NotifyProviderStatus messages.

Step 14	**end**
	
	Example:
	`Device(config-uc-wsapi)# end`
		Returns to privileged EXEC mode.

Configuring Cisco Unified Communications Gateway Services - Secure Mode

Note If the voice gateway is already configured with Cisco Unified Communications Gateway Services in nonsecure mode, remove the nonsecure mode configurations before you proceed with secure mode configuration.

You can configure Cisco Unified Communications Gateway Services in either nonsecure mode or secure mode. When you configure Cisco Unified Communications Gateway Services in secure mode, the command **ip http active-session-modules** *all* is enabled by default, irrespective of whether UC Service

APIs provisioned or not. Due to this, all the web applications are registered with NGINX proxy. This feature enables all the HTTP applications like UC Gateway Services APIs to register internally for enabling the service. However, if you configure **ip http secure-active-session-modules** *none*, then none of the web applications register with NGINX server.

To ensure that IOS applications are not enabled by default, configure the following. It explicitly enables web services for specific features of UC Gateway services:

ip http session-module-list [*module_list_name*] [*list_of_modules_to_be_registered*]

ip http secure-active-session-modules [*module_list_name*]

Example

The following is a sample configuration for secure mode. To register only WSAPI services, configure the following:

```
ip http session-module-list wsapi cisco_xmf,cisco_xcc,cisco_xsvc,cisco_xcdr
ip http secure-active-session-modules wsapi
```

To register only XCC services:

```
ip http session-module-list wsapi cisco_xcc
ip http secure-active-session-modules wsapi
```

Prerequisites

- Cisco IOS XE Everest Release 16.6.1 or later
- Application certificate ready to import in voice gateway
- Ensure that you have security and uck9 package licenses

Importing Application Certificate

Certificate is a digitally signed statement that is used to authenticate and to secure information on open networks.

When the voice gateway is behaving as a User Agent Server and receives HTTPS connection request from the application, the voice gateway requires the certificate of the application. You have to import the application certificate on to the voice gateway. By importing the application certificate, the voice gateway trusts the application, authenticates the request and establishes HTTPS connection.

Perform this procedure to import the application certificate to voice gateway.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto pki trustpoint** *trustpoint-name*
4. **enrollment terminal**
5. **exit**
6. **crypto pki authenticate** *trust-point name*
7. **Copy the application certificate and paste it on the voice gateway console**
8. **exit**

DETAILED STEP

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto pki trustpoint** *trustpoint-name* **Example:** `Device(config)#crypto pki trustpoint sampletpname`	Creates a trustpoint.
Step 4	**enrollment terminal** **Example:** `Device(ca-trustpoint)# enrollment terminal`	Specifies the manual cut-and-paste certificate enrollment method.
Step 5	**exit** **Example:** `Device(ca-trustpoint)# exit`	Exits ca-trustpoint configuration mode and returns to global configuration mode.

Step 6	**crypto pki authenticate** *name*
	
	`Device(config)# crypto pki authenticate sampletpname`
	
	`Enter the base 64 encoded CA certificate. End with a blank line or the word with "quit" on a line by itself.`
		Requests the application certificate and authenticates it.
		• The certificate request will be displayed on the console terminal so that it may be manually copied (or cut).

Step 7	**Copy the application certificate and paste it on the voice gateway console.**	Manually copy the application certificate text and paste it on the console. • Enter the base 64 encoded CA certificate. End with a blank line or the word "quit" on a line by itself.

Step 8	**end**
	
	Example:
	`Device(config)# end`
		Returns to privileged EXEC mode.

Exporting Voice Gateway Certificate to the Application

When the voice gateway is behaving as a User Agent Client and requests HTTPS connection to the application, the application requires the voice gateway certificate. You have to export the voice gateway certificate to the application. When application has the voice gateway certificate, it trusts the HTTPS requests coming from the voice gateway and establishes the HTTPS connection.

Note If no trustpoint is configured, voice gateway generates self-signed certificate and uses the same for secure communication. For more information, see the "Configuring a Trustpoint and Specifying Self-Signed Certificate Parameters" section in *Configuring Certificate Enrollment for a PKI*.

Perform this procedure to export voice gateway certificate to the application.

SUMMARY STEPS

Step 1 Execute the following command to see the voice gateway's self-signed certificate:

Example:

```
Device#show crypto pki certificates
Router Self-Signed Certificate
  Status: Available
  Certificate Serial Number (hex): 01
  Certificate Usage: General Purpose
  Issuer:
    cn=IOS-Self-Signed-Certificate-378897163
  Subject:
    Name: IOS-Self-Signed-Certificate-378897163
    cn=IOS-Self-Signed-Certificate-378897163
  Validity Date:
    start date: 12:06:27 IST Jan 18 2017
    end   date: 05:30:00 IST Jan 1 2020
  Associated Trustpoints: TP-self-signed-378897163
Storage: nvram:IOS-Self-Sig#1.cer
```

Voice gateway's self-signed certificate is shown under **`Associated Trustpoints:`**

Step 2 Execute **crypto pki export** *certificate-name* **pem terminal** command to get certificate associated with the trustpoint.

Example:

```
Device(config)#crypto pki export TP-self-signed-378897163 pem terminal
% Self-signed CA certificate:
-----BEGIN CERTIFICATE-----
MIIDLjCCAhagAwIBAgIBATANBgkqhkiG9w0BAQUFADAwMS4wLAYDVQQDEyVJT1Mt
U2VsZi1TaWduZWQtQ2VydGlmaWNhdGUtMzc4ODk3MTYzMB4XDTE3MDExODA2MzYy
N1oXDTIwMDEwMTAwMDAwMFowMDEuMCwGA1UEAxMlSU9TLVNlbGYtU2lnbmVkLUNl
cnRpZmljYXRlLTM3ODg5NzE2MzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
ggEBAI/k+Jl/RdXkUu3aBp8qIMVA7ifpRehG9AXJKlqOafc9Ly92hwNxeLGV/U8k
Xlo/fuoyaNyLIu9GwS1BfvM3yHOthhX+T5RHgcj3s1Yctl6HUW93M/EJYluo5RDE
NAXJ2UXa/Utl9ZGjCvat8h3N4QduP2ulIsK1IqyYLDRwD1fiSNFrdZB2zzIE1M7g
eeitn4n1INHiVtH0jOmO4En/FjUa3YPCFEyB1/U17YGWN/GOHguCsZluL8WywAT5
Pq1uaipVxWoCzXCb74BSxTJiHs/tmPGkIHi57RvLKxgqr5vHXCOsWsQ6/C9z6My3
tvE6dtLHuP2RgR6r+3xOhKdqcHECAwEAAaNTMFEwDwYDVR0TAQH/BAUwAwEB/zAf
BgNVHSMEGDAWgBSIzQOOrJrnxzR8LEQ2VIIfVFpO2DAdBgNVHQ4EFgQUiM0Djqya
58c0fCxENlSCH1RaTtgwDQYJKoZIhvcNAQEFBQADggEBABhYrhWv9DZ0sZZt7Smc
o5pgIIFFOtGQYc+ei7H6QNzW5iNSZbSPBAIpmVMQWHVS6cOvJ/N63ayQ+1TN3rZm
wmOU9tFExBzjge0nX+Go+0KdWNNQG4XO8SU7BKwM8iWTsM1jT1j6cb9Bv1kMgXW0
5K5AzVYTbaTP/OMoMCsuOJts+GI/Q82H7tlIbdJFbbu3iVEN+gf3coUrHa4X2jLr
K3EVLniCLedkcXdy5TppTvQM9j1FzkGMIrWAlFlp/Vh2CTigJy8GZ4pWt5QzjO6m
KuP6FZxGPNe8F5BsFCWNM5aHPa8MUqlFKZMuUb50w43SZRT3xfI2WLv1yd49f65T
mBA=
-----END CERTIFICATE-----
```

```
% General Purpose Certificate:
-----BEGIN CERTIFICATE-----
MIIDLjCCAhagAwIBAgIBATANBgkqhkiG9w0BAQUFADAwMS4wLAYDVQQDEyVJT1Mt
U2VsZi1TaWduZWQtQ2VydGlmaWNhdGUtMzc4ODk3MTYzMB4XDTE3MDExODA2MzYy
N1oXDTIwMDEwMTAwMDAwMFowMDEuMCwGA1UEAxMlSU9TLVNlbGYtU2lnbmVkLUNl
cnRpZmljYXRlLTM3ODg5NzE2MzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
ggEBAI/k+Jl/RdXkUu3aBp8qIMVA7ifpRehG9AXJKlqOafc9Ly92hwNxeLGV/U8k
Xlo/fuoyaNyLIu9GwS1BfvM3yHOthhX+T5RHgcj3s1Yctl6HUW93M/EJYluo5RDE
NAXJ2UXa/Utl9ZGjCvat8h3N4QduP2ulIsK1IqyYLDRwD1fiSNFrdZB2zzIE1M7g
eeitn4n1INHiVtH0jOmO4En/FjUa3YPCFEyB1/U17YGWN/GOHguCsZluL8WywAT5
Pq1uaipVxWoCzXCb74BSxTJiHs/tmPGkIHi57RvLKxgqr5vHXCOsWsQ6/C9z6My3
tvE6dtLHuP2RgR6r+3xOhKdqcHECAwEAAaNTMFEwDwYDVR0TAQH/BAUwAwEB/zAf
BgNVHSMEGDAWgBSIzQOOrJrnxzR8LEQ2VIIfVFpO2DAdBgNVHQ4EFgQUiM0Djqya
58c0fCxENlSCH1RaTtgwDQYJKoZIhvcNAQEFBQADggEBABhYrhWv9DZ0sZZt7Smc
o5pgIIFFOtGQYc+ei7H6QNzW5iNSZbSPBAIpmVMQWHVS6cOvJ/N63ayQ+1TN3rZm
wmOU9tFExBzjge0nX+Go+0KdWNNQG4XO8SU7BKwM8iWTsM1jT1j6cb9Bv1kMgXW0
5K5AzVYTbaTP/OMoMCsuOJts+GI/Q82H7tlIbdJFbbu3iVEN+gf3coUrHa4X2jLr
K3EVLniCLedkcXdy5TppTvQM9j1FzkGMIrWAlFlp/Vh2CTigJy8GZ4pWt5QzjO6m
KuP6FZxGPNe8F5BsFCWNM5aHPa8MUqlFKZMuUb50w43SZRT3xfI2WLv1yd49f65T
mBA=
-----END CERTIFICATE-----
```

Step 3 Copy the self-signed CA certificate displayed on the console and upload it onto the application.

Configuring Cisco Unified Communications Gateway Services in Secure Mode

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip http secure-port** *port*
4. **ip http secure-server**
5. **ip http tls-version** *version*
6. **ip http secure-trustpoint** *name*
7. **ip http max-connection** *value*
8. **ip http timeout-policy idle** *seconds* **life** *seconds* **requests** *value*
9. **http client connection persistent**
10. **http client connection idle timeout** *seconds*
11. **uc secure-wsapi**
12. **message-exchange max-failures** *number*
13. **probing max-failures** *number*
14. **probing interval keepalive** *seconds*
15. **probing interval negative** *seconds*
16. **source-address** *ip-address*
17. **end**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`enable`**
	
	Example:
	`Device> enable`
		Enables privileged EXEC mode.
		
		Enter your password if prompted.

Step 2	**`configure terminal`**
	
	Example:
	`Device# configure terminal`
		Enters global configuration mode.

Step 3	**`ip http secure-port`** *`port`*
	
	Example:
	`Device(config)# ip http secure-port 1026`
		(Optional) HTTPS connection uses inbound port 443 by default. Execute **ip http secure-port** *port* command if you want to change the default port.

Step 4	**`ip http secure-server`**
	
	Example:
	`Device(config)# ip http secure-server`
		Enables the HTTPS server (web server) on the system.
		
		Note If a certificate authority (CA) is used for certification, you should declare the CA trustpoint on the routing device before enabling the HTTPS server.
		
		Executing this command checks if there is any existing trustpoint configured.
		
		• If no trustpoint is configured, voice gateway generates self-signed certificate and uses the same for secure communication. For more information, see *Configuring Certificate Enrollment for a PKI*.
		
		• If you want to use a specific trustpoint that is already configured on the voice gateway, execute **`ip http secure-trustpoint`** <*name*> command to specify the trustpoint.

Step 5	**`ip http tls-version`** *`version`*
	
	Example:
	`Device(config)# ip http tls-version 1.2`
		(Optional) By default, all TLS versions (1.1, and 1.2) will be enabled. Configure this command if you want to use only one version of TLS.

Step 6	**`ip http secure-trustpoint`** *`name`*
	
	Example:
	`Device(config)# ip http secure-trustpoint TP-samplename`
		(Optional) Specifies the CA trustpoint that should be used to obtain certificate.
		
		• Use of this command assumes you have already declared a CA trustpoint using the **crypto ca trustpoint** command and associated submode commands.
		
		• Use the same trustpoint name that you used in the associated crypto ca trustpoint command.

	Command or Action	Purpose

Step 7	**`ip http max-connection`** *`value`*
	
	Example:
	`Device(config)# ip http max-connection 100`
		Sets the maximum number of concurrent connections to the HTTP sever that will be allowed. The default value is 5.

Step 8	**`ip http timeout-policy idle`** *`seconds`* **`life`** *`seconds`* **`requests`** *`value`*
	
	Example:
	`Device(config)# ip http timeout-policy idle 600 life 86400 requests 86400`
		Sets the characteristics that determine how long a connection to the HTTP server should remain open. The characteristics are:
		
		idle—The maximum number of seconds the connection will be kept open if no data is received or response data can not be sent out on the connection. Note that a new value may not take effect on any already existing connections. If the server is too busy or the limit on the life time or the number of requests is reached, the connection may be closed sooner. The default value is 180 seconds (3 minutes).
		
		life—The maximum number of seconds the connection will be kept open, from the time the connection is established. Note that the new value may not take effect on any already existing connections. If the server is too busy or the limit on the idle time or the number of requests is reached, it may close the connection sooner. Also, since the server will not close the connection while actively processing a request, the connection may remain open longer than the specified life time if processing is occurring when the life maximum is reached. In this case, the connection will be closed when processing finishes. The default value is 180 seconds (3 minutes). The maximum value is 86400 seconds (24 hours).
		
		requests—The maximum limit on the number of requests processed on a persistent connection before it is closed. Note that the new value may not take effect on any already existing connections. If the server is too busy or the limit on the idle time or the life time is reached, the connection may be closed before the maximum number of requests are processed. The default value is 1. The maximum value is 86400.

Step 9	**`http client connection persistent`**
	
	Example:
	`Device(config)# http client connection persistent`
		Enables HTTP persistent connection.
		
		Note When this command is configured, multiple files are loaded using the same connection. Executing this command determines whether the HTTPS client requests a keepalive or closed connection from the server. The HTTPS server is responsible for granting or denying the keepalive connection request from the client.

	Command or Action	Purpose
Step 10	**http client connection idle timeout** *seconds* **Example:** `Device(config)# http client idle timeout 600`	Sets the number of seconds that the client waits in the idle state until it closes the connection.
Step 11	**uc secure-wsapi** **Example:** `Device(config)# uc secure-wsapi`	Enters secure Cisco Unified Communications Gateway Services configuration mode.
Step 12	**message-exchange max-failures** *number* **Example:** `Device(config-uc-wsapi)# message-exchange max failures 2`	Configures the maximum number of failed message exchanges between the application and the provider before the provider stops sending messages to the application. Range is 1 to 3. Default is 1.
Step 13	**probing max-failures** *number* **Example:** `Device(config-uc-wsapi)# probing max-failures 5`	Configures the maximum number of failed probing messages before the voice gateway unregisters the application. Range is 1 to 5. Default is 3.
Step 14	**probing interval keepalive** *seconds* **Example:** `Device(config-uc-wsapi)# probing interval keepalive 180`	Configures the interval between probing messages, in seconds. Default is 120 seconds.
Step 15	**probing interval negative** *seconds* **Example:** `Device(config-uc-wsapi)# probing interval negative 10`	Configures the interval between negative probing messages, in seconds.
Step 16	**source-address** *ip-address* **Example:** `Device(config-uc-wsapi)# source-address 10.25.12.13`	Configures the IP address (hostname) as the source IP address for the Cisco Unified Communications Gateway Services. **Note** The source IP address is used by the provider in the NotifyProviderStatus messages.

Step 17	**end**
	
	Example:
	`Device(config-uc-wsapi)# end`
		Returns to privileged EXEC mode.

Configuring the XCC Provider on the Voice Gateway

Perform this procedure to configure the XCC provider on the voice gateway.

SUMMARY STEPS

1. **enable**
2. **configure terminal**

3. Enter Cisco Unified Communications Gateway Services configuration mode:
   a. **uc wsapi**

      or

   b. **uc secure-wsapi**
4. **provider xcc**
5. **no shutdown**
6. **remote-url** *url*
7. If you enable DTMF detection for XCC application and the DTMF method used for the call is **rtp-nte**, then configure the following on outbound dial-peer of CUBE:
   a. **dtmf-relay rtp-nte digit-drop**
   b. **dtmf-interworking standard**
8. **exit**
9. **end**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`enable`**
	
	Example:
	`Device> enable`
		Enables privileged EXEC mode.
		
		Enter your password if prompted.

Step 2	**`configure terminal`**
	
	Example:
	`Device# configure terminal`
		Enters global configuration mode.

Step 3	**`uc wsapi`**
	or
	`uc secure-wsapi`
	
	Example:
	`Device(config)# uc wsapi`
	
	or
	
	`Device(config)# uc secure-wsapi`
		Enters Cisco Unified Communications Gateway Services configuration mode.

Step 4	**provider xcc** **Example:** `Device(config-uc-wsapi)# provider xcc`	Enters XCC provider configuration mode.
Step 5	**no shutdown** **Example:** `Device(config-uc-wsapi-xcc)# no shutdown`	Activates XCC provider.

	Command or Action	Purpose

Step 6	**`remote-url`** *`url`*
	
	Example:
	`Device(config-uc-wsapi-xcc)# remote-url http://192.0.2.0:24/my_callcontrol`
	or
	`Device(config-uc-wsapi-xcc)# remote-url https://192.0.2.0:24/my_callcontrol`
		Specifies the URL (IP address and port number) that the application uses to communicate with XCC provider. The XCC provider uses the IP address and port to authenticate incoming requests.
		Note Only IPv4 address is allowed in secure mode (under **uc secure-wsapi** configuration).

Step 7	**`dtmf-relay rtp-nte digit-drop`**
	
	Example:
	`Device(config-uc-wsapi-xcc)# dtmf-relay rtp-nte digit-drop`
	
	`dtmf-interworking standard`
	
	Example:
	`Device(config-uc-wsapi-xcc)# dtmf-interworking standard`
		(Optional) If you enable DTMF detection for XCC application and the DTMF method used for the call is **rtp-nte**, then configure the **`dtmf-relay rtp-nte digit-drop`** and **`dtmf-interworking standard`** commands on the outbound dial-peer of CUBE to avoid any DTMF issues in the RTP data path of the call.
		Note The **digit-drop** command is available only when the **rtp-nte** keyword is configured.

Step 8	**`exit`**
	
	Example:
	`Device(config-uc-wsapi-xcc)# exit`
		Exits XCC configuration mode.

Step 9	**`end`**
	
	Example:
	`Device(config-uc-wsapi)# end`
		Returns to privileged EXEC mode.

Configuring the XSVC Provider on the Voice Gateway

Perform this procedure to configure the XSVC providers on the voice gateway.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. Enter Cisco Unified Communications Gateway Services configuration mode:
   a. **uc wsapi**
      or
   b. **uc secure-wsapi**
4. **provider xsvc**
5. **no shutdown**
6. **remote-url** [*url-number*] *url*
7. **exit**

8. **trunk group** *name*
9. **description**
10. **xsvc**
11. **exit**
12. **voip trunk group** *name*
13. **description**
14. **xsvc**
15. **session target ipv4:***destination-address*
16. **exit**
17. **end**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`enable`**
	
	Example:
	`Device> enable`
		Enables privileged EXEC mode.
		
		Enter your password if prompted.

Step 2	**`configure terminal`**
	
	Example:
	`Device# configure terminal`
		Enters global configuration mode.

Step 3	**`uc wsapi`**
	or
	`uc secure-wsapi`
	
	Example:
	`Device(config)# uc wsapi`
	
	or
	
	`Device(config)# uc secure-wsapi`
		Enters Cisco Unified Communications Gateway Services configuration mode.

Step 4	**`provider xsvc`**
	
	Example:
	`Device(config-uc-wsapi)# provider xsvc`
		Enters XSVC provider configuration mode.

Step 5	**`no shutdown`**
	
	Example:
	`Device(config-uc-wsapi-xsvc)# no shutdown`
		Activates XSVC provider.

	Command or Action	Purpose

Step 6	**remote-url** [*url-number*] *url*
	
	Example:
	`Device(config-uc-wsapi-xsvc)# remote-url 1 http://192.0.2.0:24/my_route_control`
	or
	`Device(config-uc-wsapi-xsvc)# remote-url 1 https://192.0.2.0:24/my_route_control`
		Specifies up to 8 different URLs (IP address and port number) that applications can use to communicate with the XSVC provider. The XSVC provider uses the IP address and port to authenticate incoming requests.
		The *url-number* identifies the unique url. Range is 1 to 8.
		Note In secure mode, only one remote URL is allowed. Only IPv4 address can be configured.

Step 7	**exit** **Example:** `Device(config-uc-wsapi-xsvc)# exit`	Exits XSVC configuration mode.
Step 8	**trunk group** *name* **Example:** `Device(config)# trunk group SJ_PRI`	Enters trunk-group configuration mode to define a trunk group.
Step 9	**description** **Example:** `Device(config)# description IN`	Enter a description for the trunk group. The name is passed to external application as part of XSVC status and XCC connection messages.
Step 10	**xsvc** **Example:** `Device(config-trunk-group)# xsvc`	Enables xsvc monitoring on the trunk group.
Step 11	**exit** **Example:** `Device(config-trunk-group)# exit`	Exits trunk group configuration mode.
Step 12	**voip trunk group** *name* **Example:** `Device(config)# trunk group SJ_SIP`	Enters VOIP trunk-group configuration mode to define a trunk group.
Step 13	**description** **Example:** `Device(config-voip-trk-gp)# description IN`	Enter a description for the VOIP trunk group. The name is passed to external application as part of XSVC status and XCC connection messages.
Step 14	**xsvc** **Example:** `Device(config-voip-trk-gp)# xsvc`	Enables xsvc monitoring on the VOIP trunk group.
Step 15	**session target ipv4:***destination address* **Example:** `Device(config-voip-trk-gp)# session target ipv4:9.10.31.254`	Configures the IP address of the remote voice gateway.

	Command or Action	Purpose
Step 16	**exit** **Example:** `Device(config-voip-trk-gp)# exit`	Exits VOIP trunk group configuration mode.

Step 17	**end**
	
	Example:
	`Device(config-uc-wsapi)# end`
		Returns to privileged EXEC mode.

Configuring the XCDR Provider on the Voice Gateway

Perform this procedure to configure the XCDR provider on the voice gateway.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **uc wsapi**
4. **provider xcdr**
5. **no shutdown**
6. **remote-url** [*url-number*] *url*
7. **exit**
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**uc wsapi** **Example:** `Device(config)# uc wsapi`	Enters Cisco Unified Communications Gateway Services configuration mode.
Step 4	**provider xcdr** **Example:** `Device(config-uc-wsapi)# provider xcdr`	Enters XCDR provider configuration mode.

	Command or Action	Purpose
Step 5	**no shutdown** **Example:** `Device(config-uc-wsapi-xcdr)# no shutdown`	Activates XCDR provider.
Step 6	**remote-url [***url-number***]** *url* **Example:** `Device(config-uc-wsapi-xcdr)# remote-url 1 http://209.133.85.47:8090/my_route_control`	Specifies up to eight different URLs (IP address and port number) that applications can use to communicate with the XCDR provider. The XCDR provider uses the IP address and port to authenticate incoming requests. The *url-number* identifies the unique url. Range is 1 to 8.
Step 7	**exit** **Example:** `Device(config-uc-wsapi-xcdr)# exit`	Exits XCDR configuration mode.

Step 8	**end**
	
	Example:
	`Device(config-uc-wsapi)# end`
		Returns to privileged EXEC mode.

Configuring the XMF Provider on the Voice Gateway

Perform this procedure to configure the XMF provider on the voice gateway.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **uc wsapi**
4. **provider xmf**
5. **no shutdown**
6. **remote-url** *url*
7. **exit**
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**uc wsapi** **Example:** `Device(config)# uc wsapi`	Enters Cisco Unified Communications Gateway Services configuration mode.
Step 4	**provider xmf** **Example:** `Device(config-uc-wsapi)# provider xcc`	Enters XMF provider configuration mode.
Step 5	**no shutdown** **Example:** `Device(config-uc-wsapi-xcc)# no shutdown`	Activates XMF provider.
Step 6	**remote-url** *url* **Example:** `Device(config-uc-wsapi-xcc)# remote-url http://209.133.85.47:8090/my_callcontrol`	Specifies the URL (IP address and port number) that the application uses to communicate with XMF provider. The XMF provider uses the IP address and port to authenticate incoming requests.
Step 7	**exit** **Example:** `Device(config-uc-wsapi-xcc)# exit`	Exits XMF configuration mode.

Step 8	**end**
	
	Example:
	`Device(config-uc-wsapi)# end`
		Returns to privileged EXEC mode.

Configuration Example

The following example sets up the voice gateway for Cisco Unified Communications Gateway Services. It enables the HTTP server and the XCC, XSVC, and XCDR providers. The configuration specifies the address and port that the application uses to communicate with the XCC, XSVC, and XCDR provider. It also identifies the trunk group that XSVC will be monitoring.

Note XSVC and XCDR can support up to eight different remote URLs.

```
ip http server
!
call fallback monitor
call fallback icmp-ping count 1 interval 2 timeout 100
!
uc wsapi
 source-address 10.1.1.1
 provider xcc
  remote-url http://test.com:8090/xcc
 !
 provider xsvc
  remote-url 1 http://test.com:8090/xsvc
!
 provider xcdr
  remote-url 1 http://test.com:8090/xcdr
!
trunk group pri
 xsvc

voip trunk group 1
 xsvc
 session target ipv4: 11.1.1.1
!
interface Serial0/1/0:23
isdn switch-type primary-ni
isdn incoming-voice voice
 trunk-group pri
```

Verifying and Troubleshooting Cisco Unified Communications Gateway Services

Use the following show commands to gather information on the performance of the Cisco Unified Communications Gateway Services:

- **show wsapi registration**
- **show wsapi http client**
- **show wsapi http server**
- **show wsapi xsvc routes**

Use the following debug commands to gather troubleshooting information on the service provider:

- **debug wsap**i **xcc [CR | all | function | default | detail | error | inout | event]**
- **debug wsapi xsvc [CR | all | function | default | detail | error | inout | event]**
- **debug wsapi xcdr [CR | all | function | default | detail | error | inout | event**]
- **debug wsapi xmf [CR | all | function | default | detail | error | inout | event**]
- **debug wsapi infrastructure [CR | all | function | default | detail | error | inout | event]**

Command Reference

This section documents the CLI commands that are used on the voice gateway.

- message-exchange max-failures
- probing interval
- probing max-failures
- provider
- remote-url
- show call media forking
- show voip trunk group
- show wsapi
- source-address (uc-wsapi)
- uc wsapi
- uc secure-wsapi
- voip trunk group
- xsvc

Provider and Application Interactions

This section describes the interaction and sequence of messages that take place between the providers on the voice gateway and the application.

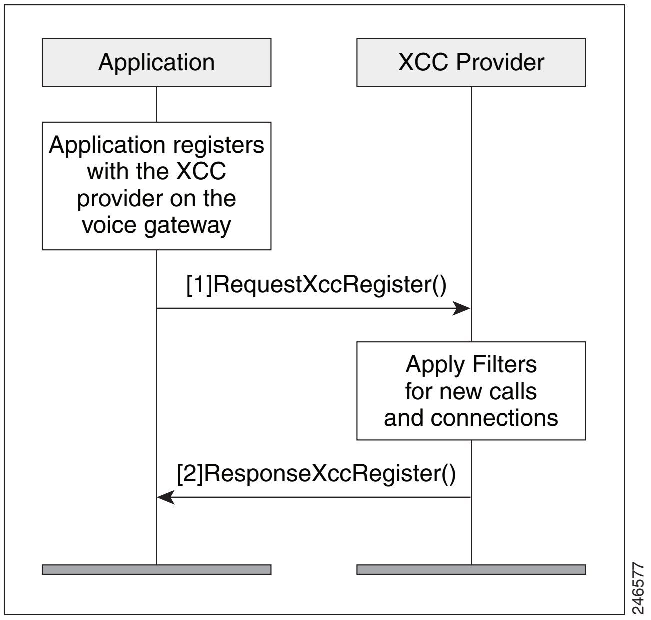
XCC

This section describes some of the interactions that takes place between the XCC provider and the application.

Interaction Between the XCC Provider and Application

Figure A-1 shows the interaction and the sequence of messages that are exchanged between the application and the XCC provider during registration.

Figure A-1 Message interaction when the application registers with XCC Provider


Application
XCC Provider
Application registers with the XCC provider on the voice gateway
[1]RequestXccRegister()
Apply Filters for new calls and connections
[2]ResponseXccRegister()
246577

Message Examples

This section provides examples of message exchanges between the application and the XCC provider.

Example of a Registration Message Exchange

The following is an example of the RequestXccRegister message sent by the application requesting registration and setting up the connection event and media event filters.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <RequestXccRegister xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <applicationData>
        <name>myapp</name>
        <url>http://test.com:8090/xcc</url>
      </applicationData>
      <blockingEventTimeoutSec>1</blockingEventTimeoutSec>
      <blockingTimeoutHandle>CONTINUE_PROCESSING</blockingTimeoutHandle>
      <connectionEventsFilter>CREATED AUTHORIZE_CALL ADDRESS_ANALYZE REDIRECTED ALERTING
CONNECTED TRANSFERRED CALL_DELIVERY DISCONNECTED HANDOFFLEAVE
HANDOFFJOIN</connectionEventsFilter>
      <mediaEventsFilter>MODE_CHANGE DTMF TONE_BUSY TONE_DIAL TONE_SECOND_DIAL
TONE_RINGBACK TONE_OUT_OF_SERVICE MEDIA_ACTIVITY</mediaEventsFilter>
      <msgHeader>
        <transactionID>11111d</transactionID>
      </msgHeader>
      <providerData>
        <url>http://10.1.1.1:8090/cisco_xcc</url>
      </providerData>
    </RequestXccRegister>
  </soapenv:Body>
</soapenv:Envelope>
```

The following is an example of a ResponseXccRegister message sent from the XCC provider in response to the application's registration request. The registration ID is used in all messages during the registered session:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <ResponseXccRegister xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>11111d</transactionID>
        <registrationID>152E034C:XCC:myapp:5</registrationID>
      </msgHeader>
      <providerStatus>IN_SERVICE</providerStatus>
    </ResponseXccRegister>
  </SOAP:Body>
</SOAP:Envelope>
I/O warning : failed to load external entity "ResponseXCCRegister.txt"
mcebu-reg-ex2:428>  xmllint --format xmlResponseXCCRegister > ResponseXCCRegister.txt
mcebu-reg-ex2:429> more ResponseXCCRegister.txt
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <ResponseXccRegister xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>11111d</transactionID>
        <registrationID>152E034C:XCC:myapp:5</registrationID>
      </msgHeader>
      <providerStatus>IN_SERVICE</providerStatus>
    </ResponseXccRegister>
  </SOAP:Body>
</SOAP:Envelope
```

Example of a Change in Service Message

The following is an example of a NotifyXccStatus message sent from the gateway when the XCC shuts down.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <NotifyXccProviderStatus xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>43257C78:F4</transactionID>
      </msgHeader>
      <applicationData>
        <url>http://mcebu-reg-ex2.cisco.com:8090/xcc</url>
      </applicationData>
      <providerData>
        <url>http://172.19.149.185:8090/cisco_xcc</url>
      </providerData>
      <providerStatus>SHUTDOWN</providerStatus>
    </NotifyXccProviderStatus>
  </SOAP:Body>
</SOAP:Envelope>
```

Example of the Application Requesting to be Unregister

The following is an example of a RequestXccUnRegister message sent from an application when it no longer needs the provider's services.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <SolicitXccProviderUnRegister xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>152EF0C4:8F</transactionID>
        <registrationID>152E034C:XCC:myapp:5</registrationID>
      </msgHeader>
    </SolicitXccProviderUnRegister>
  </SOAP:Body>
</SOAP:Envelope>
```

Example of a Keepalive Probing Message

The following is an example of the SolicitXccProbing message sent from the XCC provider to maintain an active registration session.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <SolicitXccProbing xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>152EC69C:89</transactionID>
        <registrationID>152E034C:XCC:myapp:5</registrationID>
      </msgHeader>
      <sequence>1</sequence>
      <interval>5</interval>
      <failureCount>0</failureCount>
      <registered>true</registered>
      <providerStatus>IN_SERVICE</providerStatus>
    </SolicitXccProbing>
  </SOAP:Body>
</SOAP:Envelope>
```

The following is an example of the ResponseXccProbing message sent from the application responding to the XCC provider probing message.

```
<?xml version="1.0"?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Header/>
  <soapenv:Body>
    <ResponseXccProbing xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <registrationID>62199E50:XCC:myapp:34</registrationID>
        <transactionID>621B7310:346</transactionID>
      </msgHeader>
      <sequence>1</sequence>
    </ResponseXccProbing>
  </soapenv:Body>
</soapenv:Envelope>
```

Example of the Provider Shutting Down

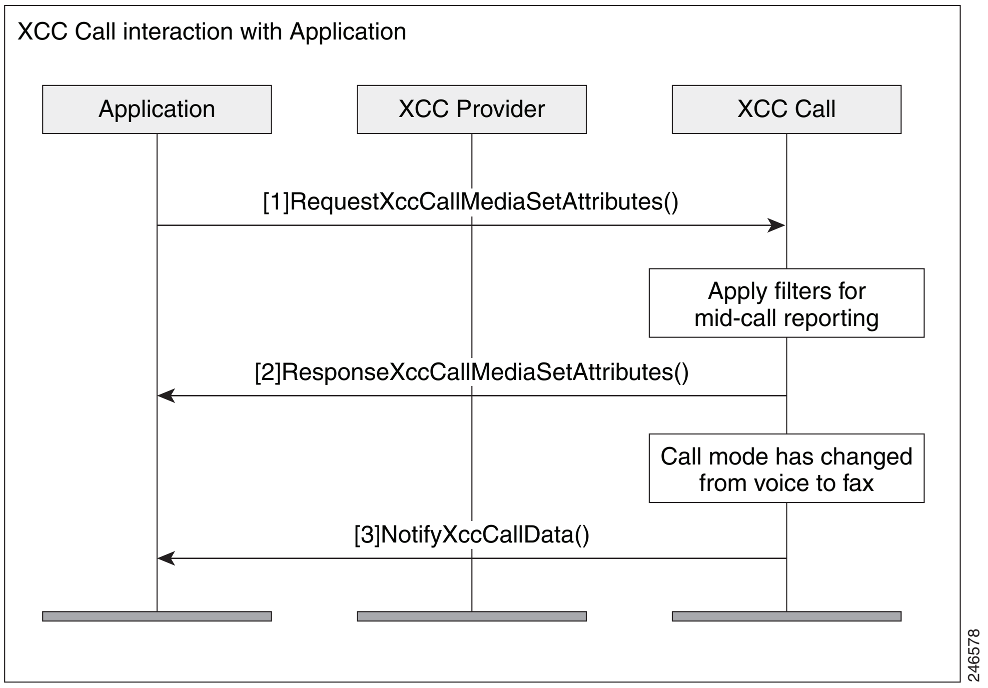
The following is an example of the SolicitXccProviderUnRegister message sent from the XCC provider when it enters the shutdown state.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <SolicitXccProviderUnRegister xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>152EF0C4:8F</transactionID>
        <registrationID>152E034C:XCC:myapp:5</registrationID>
      </msgHeader>
    </SolicitXccProviderUnRegister>
  </SOAP:Body>
</SOAP:Envelope>
```

Interaction Between the Application, XCC Provider, and XCC Call

Figure A-2 shows the interaction between the application, XCC provider, and XCC call for a call and the sequence of messages that are exchanged between the application and the XCC provider.

Figure A-2 Message interaction when a call comes in


XCC Call interaction with Application
Application
XCC Provider
XCC Call
[1]RequestXccCallMediaSetAttributes()
Apply filters for mid-call reporting
[2]ResponseXccCallMediaSetAttributes()
Call mode has changed from voice to fax
[3]NotifyXccCallData()
246578

Message Examples

This section provides examples of message exchanges between the application and the XCC provider during a call.

Example of the Application Setting Call Media Attributes.

The following is an example of a RequestXccCallMediaSetAttributes message sent from application notifying the provider of the media attributes for a call.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <RequestXccCallMediaSetAttributes xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <callID>6</callID>
      <mediaForking>
        <farEndAddr>
          <ipv4>1.3.45.155</ipv4>
          <port>32599</port>
        </farEndAddr>
        <nearEndAddr>
          <ipv4>1.3.45.155</ipv4>
          <port>32598</port>
        </nearEndAddr>
      </mediaForking>
      <msgHeader>
        <registrationID>D3868:XCC:myapp:5</registrationID>
        <transactionID>D5494:5B</transactionID>
      </msgHeader>
    </RequestXccCallMediaSetAttributes>
  </soapenv:Body>
</soapenv:Envelope>
```

The following is an example of the ResponseXccCallMediaSetAttributes message sent from the a XCC provider in response to the application's media set attribute request.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <ResponseXccCallMediaSetAttributes xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>D5494:5B</transactionID>
        <registrationID>D3868:XCC:myapp:5</registrationID>
      </msgHeader>
    </ResponseXccCallMediaSetAttributes>
  </SOAP:Body>
</SOAP:Envelope>
```

Example of a Change in Call Mode

The following is an example of a NotifyXccCallData message sent from the XCC provider notifying the application that the call mode has changed from modem to fax mode.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http:// www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <NotifyXccCallData>
      <msgHeader>
        <transactionID>2336EF94:BC</transactionID>
        <registrationID>23362C88:XCC:myapp:7</registrationID>
      </msgHeader>
      <callData>
        <callID>8</callID>
        <state>ACTIVE</state>
      </callData>
      <mediaEvent>
        <modeChange>
          <old>MODEM</old>
          <new>FAX</new>
        </modeChange>
      </mediaEvent>
    </NotifyXccCallData>
  </SOAP:Body>
</SOAP:Envelope>
```

Example of a DTMF Detection

The following is an example of a NotifyXccCallData message sent from the XCC provider notifying the application that the number 1 digit on the keypad has been pressed.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <NotifyXccCallData>
      <msgHeader>
        <transactionID>491100E4:2E5</transactionID>
        <registrationID>4910E328:XCC:myapp:29</registrationID>
      </msgHeader>
      <callData>
        <callID>38</callID>
        <state>ACTIVE</state>
      </callData>
      <mediaEvent>
        <DTMF>
          <digit>1</digit>
          <dateTime>*01:35:04.111 UTC Sun Oct 4 1970</dateTime>
        </DTMF>
      </mediaEvent>
    </NotifyXccCallData>
  </SOAP:Body>
</SOAP:Envelope>
```

Example of Call Media Forking

The following is an example of a RequestXccCallMediaForking message sent from the application requesting that the media stream for the call session be forked. The application must include two unique RTP ports—nearEndAddr element for the forked TX media stream and the farEndAddr XCC element for the RX media stream

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <RequestXccCallMediaForking xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <action>
        <enableMediaForking>
          <farEndAddr>
            <ipv4>1.3.45.155</ipv4>
            <port>32599</port>
          </farEndAddr>
          <nearEndAddr>
            <ipv4>1.3.45.155</ipv4>
            <port>32598</port>
          </nearEndAddr>
        </enableMediaForking>
      </action>
      <callID>8</callID>
      <msgHeader>
        <registrationID>4C21504:XCC:myapp:3</registrationID>
        <transactionID>4C23C6C:2FE</transactionID>
      </msgHeader>
    </RequestXccCallMediaForking>
  </soapenv:Body>
</soapenv:Envelope>
```

The following is an example of the NotifyXccCallData message sent from the XCC provider to the application with information on the status of the media forking.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <NotifyXccCallData>
      <msgHeader>
        <transactionID>4C252A4:2FF</transactionID>
        <registrationID>4C21504:XCC:myapp:3</registrationID>
      </msgHeader>
      <callData>
        <callID>8</callID>
        <state>ACTIVE</state>
      </callData>
      <mediaEvent>
        <mediaForking>
          <mediaForkingState>STARTED</mediaForkingState>
        </mediaForking>
      </mediaEvent>
    </NotifyXccCallData>
  </SOAP:Body>
</SOAP:Envelope>
```

The following is an example of the ResponseXccCallMediaForking message sent from the XCC provider in response to the application's media forking request.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <ResponseXccCallMediaForking xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>4C23C6C:2FE</transactionID>
        <registrationID>4C21504:XCC:myapp:3</registrationID>
      </msgHeader>
    </ResponseXccCallMediaForking>
  </SOAP:Body>
</SOAP:Envelope
```

Interaction Between the Application and XCC Connection

The following section describes the interaction between the application, XCC provider and XCC Connection.

Examples of XCC Message Exchange in the Connection State

The following is an example of a notification message sent from the XCC provider notifying the application of a connection creation event.

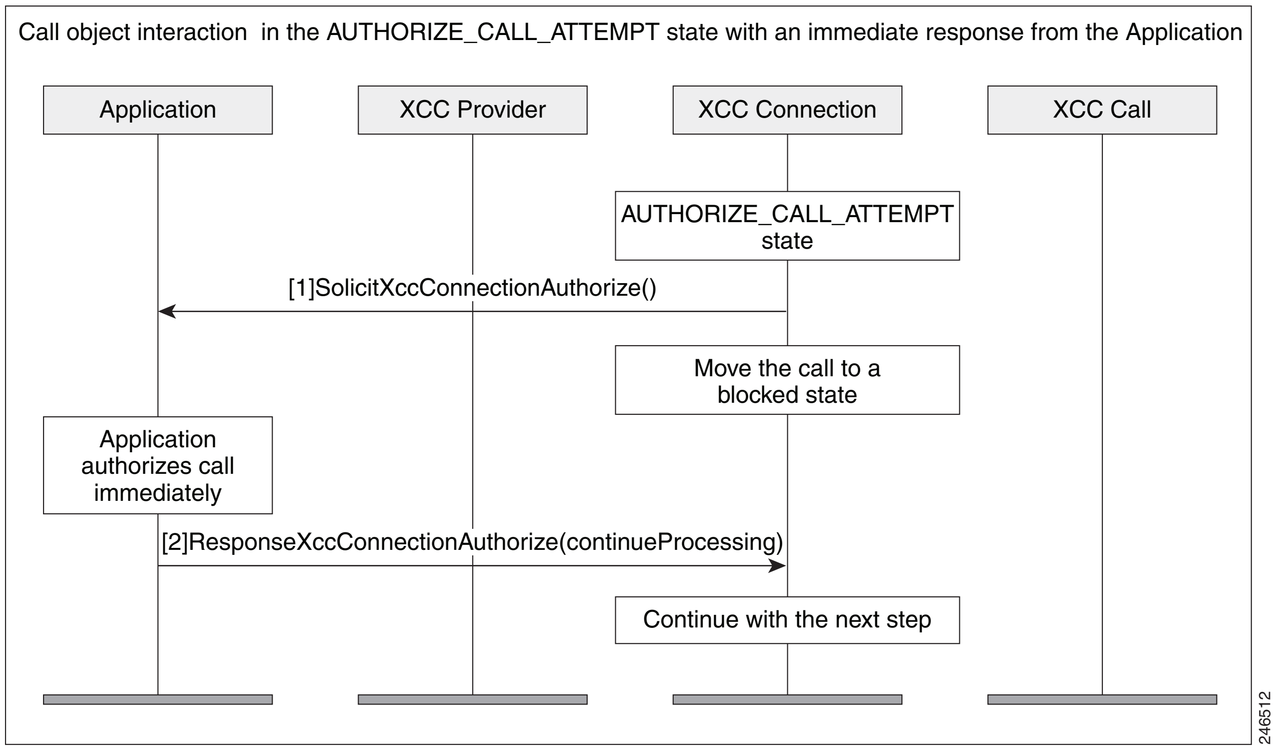
```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <NotifyXccConnectionData xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>152E6854:69</transactionID>
        <registrationID>152E034C:XCC:myapp:5</registrationID>
      </msgHeader>
      <callData>
        <callID>5</callID>
        <state>ACTIVE</state>
      </callData>
      <connData>
        <connID>30</connID>
        <state>IDLE</state>
      </connData>
      <event>
        <created>
          <connDetailData>
            <connData>
              <connID>30</connID>
              <state>IDLE</state>
            </connData>
            <guid>DDAAE040-7F44-11E0-831A-D2E9BAD25129</guid>
            <callingAddrData>
              <type>E164</type>
              <addr>3901</addr>
            </callingAddrData>
            <calledAddrData>
              <type>E164</type>
              <addr>2002</addr>
            </calledAddrData>
            <origCallingAddrData>
              <type>E164</type>
```

```
                <addr>3901</addr>
              </origCallingAddrData>
              <origCalledAddrData>
                <type>E164</type>
                <addr>2002</addr>
              </origCalledAddrData>
              <connIntfType>CONN_SIP</connIntfType>
              <mediaData>
                <type>VOICE</type>
              </mediaData>
              <connIntf>9.10.31.254</connIntf>
              <routeName>SANJOSE_SIP</routeName>
              <routeDescription>IN</routeDescription>
              <connDirectionType>INCOMING</connDirectionType>
            </connDetailData>
          </created>
        </event>
      </NotifyXccConnectionData>
    </SOAP:Body>
  </SOAP:Envelope>
```

Interaction for Call Authorization with an Immediate Response

Figure A-3 illustrates the call interaction when an application responds immediately to a call authorization solicit message from the XCC provider.

Figure A-3 Call Interaction when the application responds immediately to a call


Call object interaction in the AUTHORIZE_CALL_ATTEMPT state with an immediate response from the Application
Application
XCC Provider
XCC Connection
XCC Call
AUTHORIZE_CALL_ATTEMPT state
[1]SolicitXccConnectionAuthorize()
Move the call to a blocked state
Application authorizes call immediately
[2]ResponseXccConnectionAuthorize(continueProcessing)
Continue with the next step
246512

The following example is the SolicitXccConnectionAuthorize message sent from the XCC provider asking for authorization to continue processing the call.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <SolicitXccConnectionAuthorize xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>152E6854:6A</transactionID>
        <registrationID>152E034C:XCC:myapp:5</registrationID>
      </msgHeader>
      <callData>
        <callID>5</callID>
        <state>ACTIVE</state>
      </callData>
      <connDetailData>
        <connData>
          <connID>30</connID>
          <state>AUTHORIZE_CALL_ATTEMPT</state>
        </connData>
        <guid>DDAAE040-7F44-11E0-831A-D2E9BAD25129</guid>
        <callingAddrData>
          <type>E164</type>
          <addr>3901</addr>
        </callingAddrData>
        <calledAddrData>
          <type>E164</type>
          <addr>2002</addr>
        </calledAddrData>
        <origCallingAddrData>
          <type>E164</type>
          <addr>3901</addr>
        </origCallingAddrData>
        <origCalledAddrData>
          <type>E164</type>
          <addr>2002</addr>
        </origCalledAddrData>
        <connIntfType>CONN_SIP</connIntfType>
        <mediaData>
          <type>VOICE</type>
        </mediaData>
        <connIntf>9.10.31.254</connIntf>
            <routeName>SANJOSE_SIP</routeName>
            <routeDescription>IN</routeDescription>
        <connDirectionType>INCOMING</connDirectionType>
      </connDetailData>
    </SolicitXccConnectionAuthorize>
  </SOAP:Body>
</SOAP:Envelope>
```

Upon authentication, the application immediately sends a response. The following example is the response message (ResponseXccConnectionAuthorize) from the application to continue processing the call.

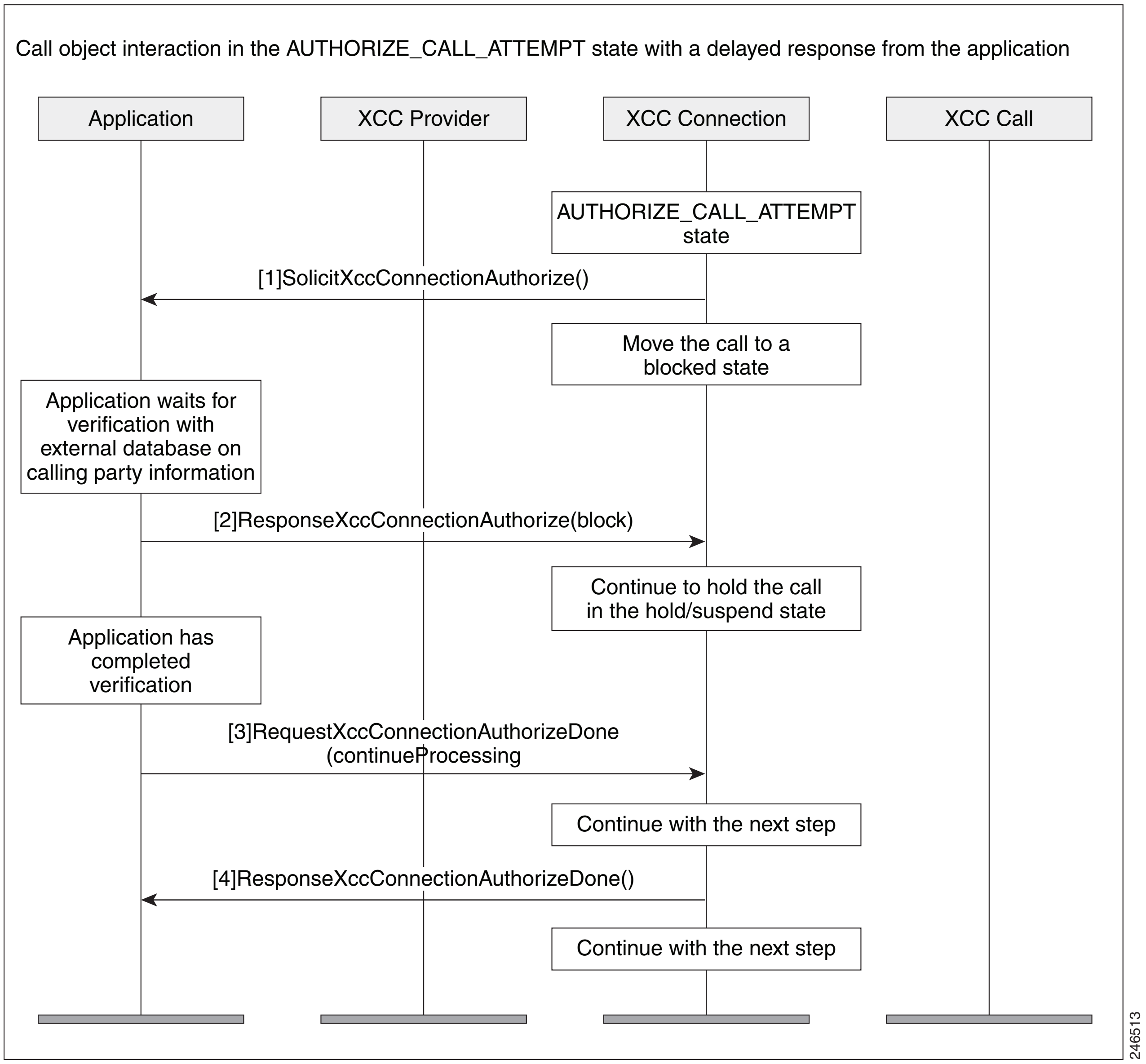
```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Header/>
  <soapenv:Body>
    <ResponseXccConnectionAuthorize xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <action>continueProcessing</action>
      <msgHeader>
        <registrationID>152E034C:XCC:myapp:5</registrationID>
        <transactionID>152E6854:6A</transactionID>
      </msgHeader>
```

```
        </ResponseXccConnectionAuthorize>
      </soapenv:Body>
    </soapenv:Envelope>
```

Interaction for Call Authorization with a Delayed Response

Figure A-4 illustrates the call interaction when an application cannot respond immediately to a call authorization solicit message from the XCC provider. The application can request that the XCC provider temporarily block the call.

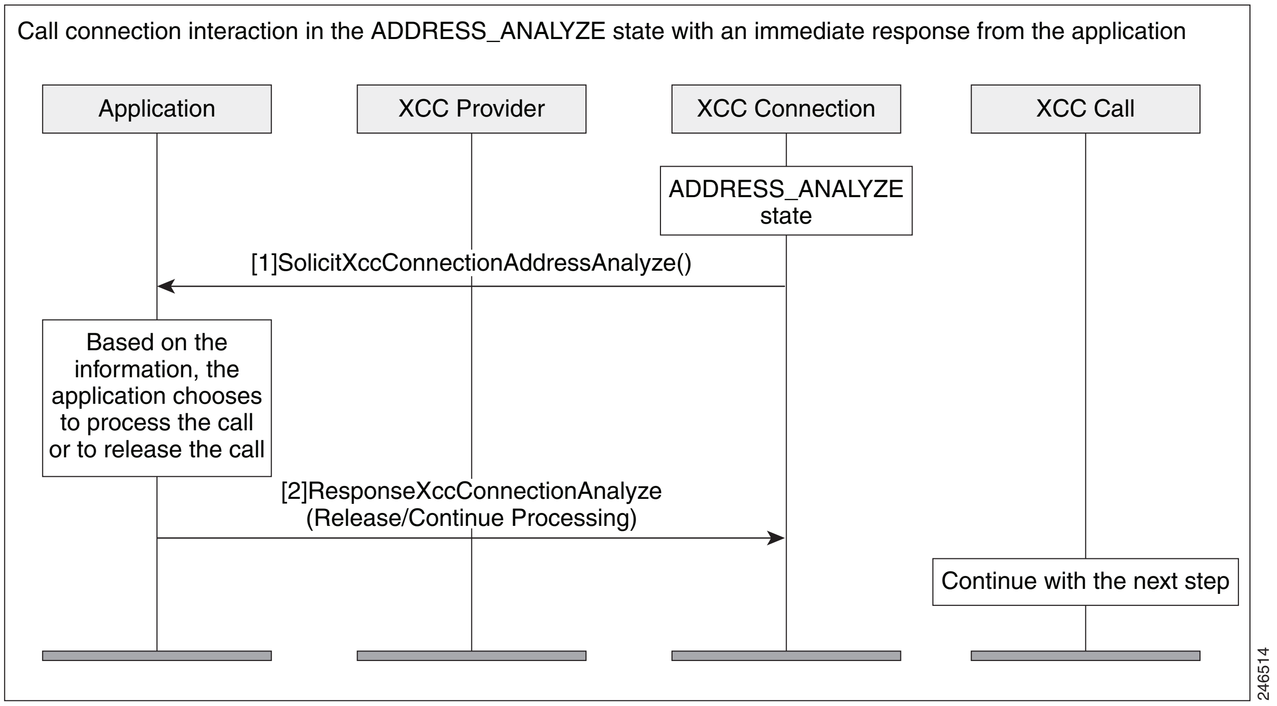
Figure A-4 Call Interaction when the application has a delayed response


Call object interaction in the AUTHORIZE_CALL_ATTEMPT state with a delayed response from the application
Application
XCC Provider
XCC Connection
XCC Call
AUTHORIZE_CALL_ATTEMPT state
[1]SolicitXccConnectionAuthorize()
Move the call to a blocked state
Application waits for verification with external database on calling party information
[2]ResponseXccConnectionAuthorize(block)
Continue to hold the call in the hold/suspend state
Application has completed verification
[3]RequestXccConnectionAuthorizeDone (continueProcessing
Continue with the next step
[4]ResponseXccConnectionAuthorizeDone()
Continue with the next step
246513

Interaction During Digit Collection with an Immediate Response

Figure A-5 shows the call interaction after an application has sent a message to the XCC provider to continue the call and begin collecting digits. The application is able to respond immediately.

Figure A-5 Call Interaction when the application responds immediately upon digit collection


Call connection interaction in the ADDRESS_ANALYZE state with an immediate response from the application
Application
XCC Provider
XCC Connection
XCC Call
ADDRESS_ANALYZE state
[1]SolicitXccConnectionAddressAnalyze()
Based on the information, the application chooses to process the call or to release the call
[2]ResponseXccConnectionAnalyze (Release/Continue Processing)
Continue with the next step
246514

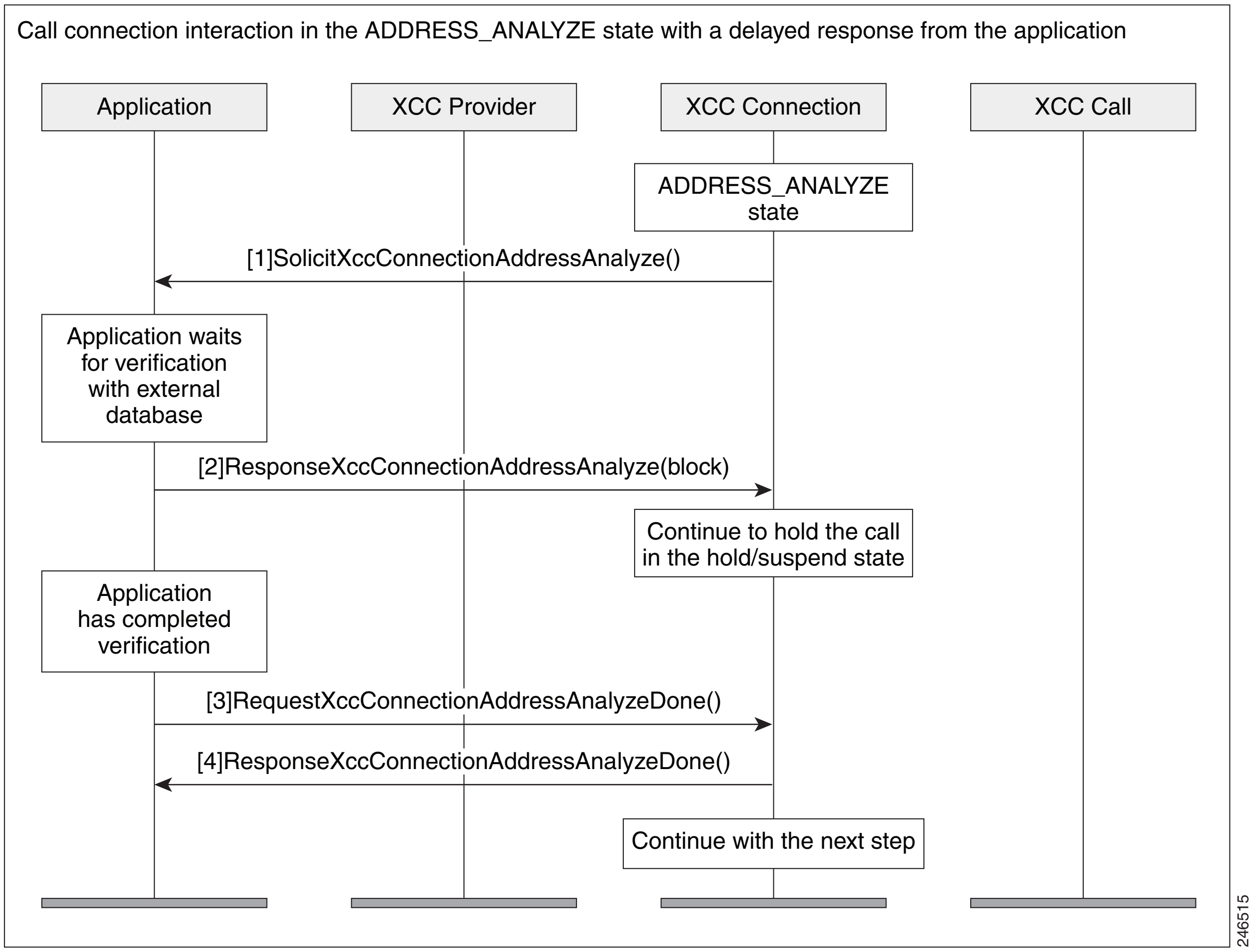
The following example is the SolicitXccConnectionAddressAnalyze message sent from the XCC provider with call information for the application.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <SolicitXccConnectionAddressAnalyze
xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>152E6B18:6B</transactionID>
        <registrationID>152E034C:XCC:myapp:5</registrationID>
      </msgHeader>
      <callData>
        <callID>5</callID>
        <state>ACTIVE</state>
      </callData>
      <connData>
        <connID>30</connID>
        <state>ADDRESS_ANALYZE</state>
      </connData>
      <collectAddress>
        <type>E164</type>
        <addr>2002</addr>
      </collectAddress>
    </SolicitXccConnectionAddressAnalyze>
  </SOAP:Body>
</SOAP:Envelope>
```

Interaction During Digit Collection with a Delayed Response

Figure A-6 shows the call interaction after an application has sent a message to the XCC provider to continue to begin collecting digits, but the application is unable to respond immediately.

Figure A-6 Call Interaction when the application has a delayed response to digit collections


Call connection interaction in the ADDRESS_ANALYZE state with a delayed response from the application
Application
XCC Provider
XCC Connection
XCC Call
ADDRESS_ANALYZE state
[1]SolicitXccConnectionAddressAnalyze()
Application waits for verification with external database
[2]ResponseXccConnectionAddressAnalyze(block)
Continue to hold the call in the hold/suspend state
Application has completed verification
[3]RequestXccConnectionAddressAnalyzeDone()
[4]ResponseXccConnectionAddressAnalyzeDone()
Continue with the next step
246515

Notification Examples

The following example is the NotifyXccConnection message sent from the XCC provider letting the application know that an outgoing call is being connected.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <NotifyXccConnectionData xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>490D843C:2C9</transactionID>
        <registrationID>490D710C:XCC:myapp:28</registrationID>
      </msgHeader>
      <callData>
        <callID>37</callID>
        <state>ACTIVE</state>
      </callData>
      <connData>
        <connID>280</connID>
        <state>CONNECTED</state>
      </connData>
      <event>
        <CONNECTED>
          <connDetailData>
            <connData>
              <connID>280</connID>
              <state>CONNECTED</state>
            </connData>
            <guid>B64EC537-872C-11E0-8FBC-A55F7D9A8E13</guid>
            <callingAddrData>
              <type>E164</type>
              <addr>2001</addr>
            </callingAddrData>
            <calledAddrData>
              <type>E164</type>
              <addr>3001</addr>
            </calledAddrData>
            <origCallingAddrData>
              <type>E164</type>
              <addr>2001</addr>
            </origCallingAddrData>
            <origCalledAddrData>
              <type>E164</type>
              <addr>3001</addr>
            </origCalledAddrData>
            <connIntfType>CONN_SIP</connIntfType>
            <mediaData>
              <type>VOICE</type>
              <coderType>g711ulaw</coderType>
              <coderByte>160</coderByte>
            </mediaData>
            <connIntf>9.10.21.254</connIntf>
            <connDirectionType>INCOMING</connDirectionType>
          </connDetailData>
        </CONNECTED>
      </event>
    </NotifyXccConnectionData>
  </SOAP:Body>
</SOAP:Envelope>
```

The following example is the NotifyXccConnection message sent from the XCC provider letting the application know that a transferred event has occurred.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <NotifyXccConnectionData xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>48EE6610:2B2</transactionID>
        <registrationID>48EDDDC8:XCC:myapp:27</registrationID>
      </msgHeader>
      <callData>
        <callID>36</callID>
        <state>ACTIVE</state>
      </callData>
      <connData>
        <connID>274</connID>
        <state>DISCONNECTED</state>
      </connData>
      <event>
        <disconnected>
```

```
          <mediaData>
            <type>VOICE</type>
            <coderType>g711ulaw</coderType>
            <coderByte>160</coderByte>
          </mediaData>
          <statsData>
            <callDuration>PT7.46S</callDuration>
            <TxPacketsCount>365</TxPacketsCount>
            <TxBytesCount>58400</TxBytesCount>
            <TxDurationMSec>0</TxDurationMSec>
            <TxVoiceDurationMSec>0</TxVoiceDurationMSec>
            <RxPacketsCount>359</RxPacketsCount>
            <RxBytesCount>57440</RxBytesCount>
            <RxDurationMSec>0</RxDurationMSec>
            <RxVoiceDurationMSec>0</RxVoiceDurationMSec>
          </statsData>
          <discCause>16</discCause>
          <jitterData>
            <routeTripDelayMSec>0</routeTripDelayMSec>
            <onTimeRvPlayMSec>0</onTimeRvPlayMSec>
            <gapFillWithPredicationMSec>0</gapFillWithPredicationMSec>
            <gapFillWithInterpolationMSec>0</gapFillWithInterpolationMSec>
            <gapFillWithRedundancyMSec>0</gapFillWithRedundancyMSec>
            <lostPacketsCount>0</lostPacketsCount>
            <earlyPacketsCount>0</earlyPacketsCount>
            <latePacketsCount>0</latePacketsCount>
            <receiveDelayMSec>0</receiveDelayMSec>
            <loWaterPlayoutDelayMSec>0</loWaterPlayoutDelayMSec>
            <hiWaterPlayoutDelayMSec>0</hiWaterPlayoutDelayMSec>
          </jitterData>
        </disconnected>
      </event>
    </NotifyXccConnectionData>
  </SOAP:Body>
</SOAP:Envelope>
```

The following example is the NotifyXccConnection message sent from the XCC provider letting the application know that a transfer handoff leave event has occurred.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <NotifyXccConnectionData xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>48EE65AC:2AC</transactionID>
        <registrationID>48EDDDC8:XCC:myapp:27</registrationID>
      </msgHeader>
      <callData>
        <callID>35</callID>
        <state>ACTIVE</state>
      </callData>
      <connData>
        <connID>272</connID>
        <state>CONNECTED</state>
      </connData>
      <event>
        <handoffLeave/>
      </event>
    </NotifyXccConnectionData>
  </SOAP:Body>
</SOAP:Envelope>
```

The following example is the NotifyXccConnection message sent from the XCC provider letting the application know that a transfer handoff join event has occurred.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <NotifyXccConnectionData xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0">
      <msgHeader>
        <transactionID>48EE65AC:2AD</transactionID>
        <registrationID>48EDDDC8:XCC:myapp:27</registrationID>
      </msgHeader>
      <callData>
        <callID>36</callID>
        <state>ACTIVE</state>
      </callData>
      <connData>
        <connID>272</connID>
        <state>CONNECTED</state>
      </connData>
      <event>
        <handoffJoin>
          <connDetailData>
            <connData>
              <connID>272</connID>
              <state>CONNECTED</state>
            </connData>
            <guid>99CFA037-F5F2-11B2-8255-AC403F9877FF</guid>
            <callingAddrData>
              <type>E164</type>
              <addr>2001</addr>
            </callingAddrData>
            <calledAddrData>
              <type>E164</type>
              <addr>3001</addr>
            </calledAddrData>
            <origCallingAddrData>
              <type>E164</type>
              <addr>2001</addr>
            </origCallingAddrData>
            <origCalledAddrData>
              <type>E164</type>
              <addr>3001</addr>
            </origCalledAddrData>
            <connIntfType>CONN_SIP</connIntfType>
            <mediaData>
              <type>VOICE</type>
              <coderType>g711ulaw</coderType>
              <coderByte>160</coderByte>
            </mediaData>
            <connIntf>9.10.31.254</connIntf>
            <routeName>SANJOSE_SIP</routeName>
            <routeDescription>IN</routeDescription>
            <connDirectionType>OUTGOING</connDirectionType>
          </connDetailData>
        </handoffJoin>
      </event>
    </NotifyXccConnectionData>
  </SOAP:Body>
</SOAP:Envelope>
```

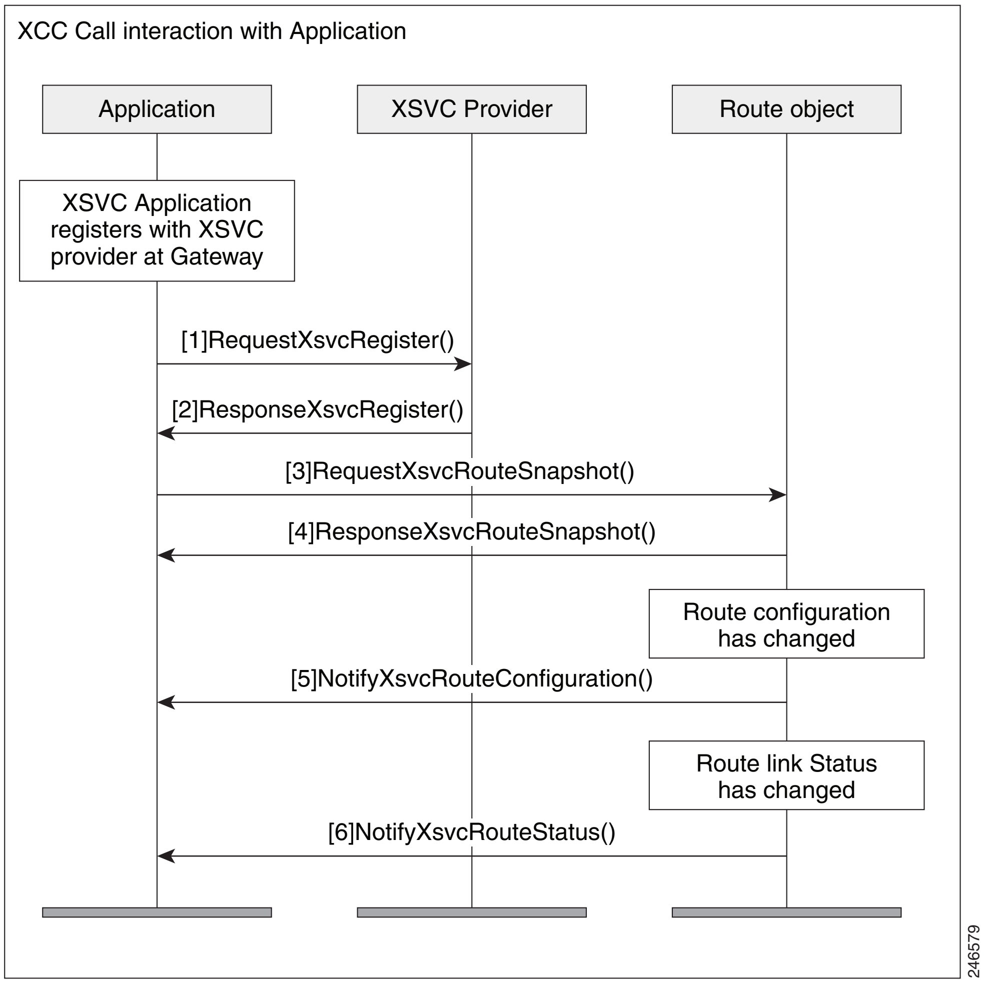
XSVC

This section describes the some of the interactions that take place between the XSVC provider and the application.

Interaction Between the XSVC Provider, Application, and Route Object

Figure A-7 shows the interaction and the sequence of messages that are exchanged between the applicatio, XSVC provider, and the route object during registration.

Figure A-7 Interaction between an applicaton , XSVC provider, and route object


XCC Call interaction with Application
Application
XSVC Provider
Route object
XSVC Application registers with XSVC provider at Gateway
[1]RequestXsvcRegister()
[2]ResponseXsvcRegister()
[3]RequestXsvcRouteSnapshot()
[4]ResponseXsvcRouteSnapshot()
Route configuration has changed
[5]NotifyXsvcRouteConfiguration()
Route link Status has changed
[6]NotifyXsvcRouteStatus()
246579

Message Examples

This section provides examples of message exchanges between the application and the XSVC provider.

Example of a Registration Message Exchange

The following is an example of a RequestXsvcRegister message sent from the application requesting registration and setting route event filters.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <RequestXsvcRegister xmlns="http://www.cisco.com/schema/cisco_xsvc/v1_0">
      <applicationData>
        <name>myapp</name>
        <url>http://test.com:8090/xsvc</url>
      </applicationData>
      <msgHeader>
        <transactionID>txID001</transactionID>
      </msgHeader>
      <providerData>
        <url>http://10.1.1.1:8090/cisco_xsvc</url>
      </providerData>
      <routeEventsFilter>ROUTE_CONF_UPDATED ROUTE_STATUS_UPDATED</routeEventsFilter>
    </RequestXsvcRegister>
  </soapenv:Body>
</soapenv:Envelope>
```

The following is an example of a ResponseXsvcRegister message sent from the XSVC provider in response to the application's registration request.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <ResponseXsvcRegister xmlns="http://www.cisco.com/schema/cisco_xsvc/v1_0">
      <msgHeader>
        <transactionID>txID001</transactionID>
        <registrationID>2F2EEC:XSVC:myapp:1</registrationID>
      </msgHeader>
      <providerStatus>IN_SERVICE</providerStatus>
    </ResponseXsvcRegister>
  </SOAP:Body>
</SOAP:Envelope>
```

The following is an example of a NotifyXsvcStatusmessage sent from the XSVC provider when it enters the shutdown state.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <NotifyXsvcProviderStatus xmlns="http://www.cisco.com/schema/cisco_xsvc/v1_0">
      <msgHeader>
        <transactionID>6A89EC:B</transactionID>
      </msgHeader>
      <applicationData>
        <url>http://test.com:8090/xsvc</url>
      </applicationData>
      <providerData>
        <url>http://10.1.1.1:8090/cisco_xsvc</url>
      </providerData>
      <providerStatus>SHUTDOWN</providerStatus>
    </NotifyXsvcProviderStatus>
  </SOAP:Body>
</SOAP:Envelope>
```

Example of a Snapshot Reponse Message

The following is an example of a ResponseXsvcRouteSnapshot message sent from XSVC provider with route information.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
```

```
  <SOAP:Body>
    <ResponseXsvcRouteSnapshot xmlns="http://www.cisco.com/schema/cisco_xsvc/v1_0">
      <msgHeader>
        <transactionID>txID002</transactionID>
        <registrationID>77F9EC:XSVC:myapp:5</registrationID>
      </msgHeader>
      <routeList>
        <route>
          <routeName>pri</routeName>
          <routeType>PSTN</routeType>
          <trunkList>
            <trunkData>
              <name>Se0/1/0:23</name>
              <type>ISDN_PRI</type>
              <status>UP</status>
            </trunkData>
          </trunkList>
        </route>
        <route>
          <routeName>1</routeName>
          <routeType>VOIP</routeType>
          <trunkList>
            <trunkData>
              <name>11.1.1.1</name>
              <type>H323</type>
              <status>UP</status>
            </trunkData>
          </trunkList>
        </route>
      </routeList>
    </ResponseXsvcRouteSnapshot>
  </SOAP:Body>
</SOAP:Envelope>
```

Example of a Route Configuration Change

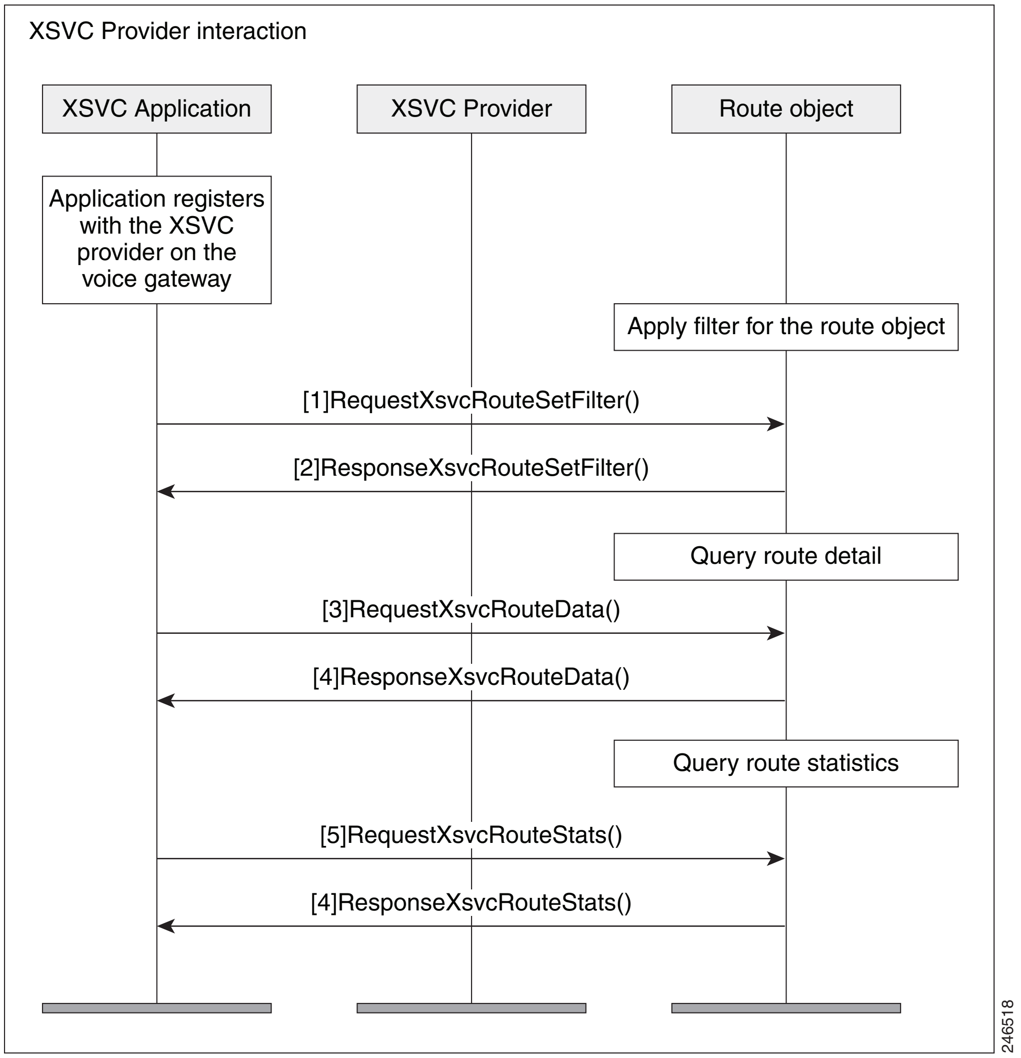
The following is an example of a NotifyXsvcRouteConfiguration message sent from XSVC provider notifying the application that the route list has been modified.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <NotifyXsvcRouteConfiguration xmlns="http://www.cisco.com/schema/cisco_xsvc/v1_0">
      <msgHeader>
        <transactionID>7FFC8C:1C</transactionID>
        <registrationID>7E4130:XSVC:myapp:6</registrationID>
      </msgHeader>
      <type>MODIFIED</type>
      <routeList>
        <route>
          <routeName>pri</routeName>
          <routeType>PSTN</routeType>
          <trunkList>
            <trunkData>
              <name>Se0/1/0:23</name>
              <type>ISDN_PRI</type>
              <status>UP</status>
            </trunkData>
          </trunkList>
        </route>
      </routeList>
    </NotifyXsvcRouteConfiguration>
  </SOAP:Body>
</SOAP:Envelope>
```

Interaction between the Application and the XSVC Provider

Figure A-8 illustrates the call interaction when an application responds immediately to a call authorization solicit message from the XSVC provider.

Figure A-8 Interaction between the application, XSVC provider, and route object when new filters are applied


XSVC Provider interaction
XSVC Application
XSVC Provider
Route object
Application registers with the XSVC provider on the voice gateway
Apply filter for the route object
[1]RequestXsvcRouteSetFilter()
[2]ResponseXsvcRouteSetFilter()
Query route detail
[3]RequestXsvcRouteData()
[4]ResponseXsvcRouteData()
Query route statistics
[5]RequestXsvcRouteStats()
[4]ResponseXsvcRouteStats()
246518

Example of a Route Data Message

The following is an example of a ResponseXsvcRouteStats message sent from XSVC provider with route statistics.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <ResponseXsvcRouteStats xmlns="http://www.cisco.com/schema/cisco_xsvc/v1_0">
```

```
      <msgHeader>
        <transactionID>txID003</transactionID>
        <registrationID>77F9EC:XSVC:myapp:5</registrationID>
      </msgHeader>
      <routeList>
        <route>
          <routeName>pri</routeName>
          <routeType>PSTN</routeType>
          <trunkList>
            <trunkData>
              <name>Se0/1/0:23</name>
              <type>ISDN_PRI</type>
              <status>UP</status>
              <currentStatics>
                <LCV>0</LCV>
                <PCV>0</PCV>
                <CSS>0</CSS>
                <SEFS>0</SEFS>
                <LES>0</LES>
                <DM>0</DM>
                <ES>0</ES>
                <BES>0</BES>
                <SES>0</SES>
                <UAS>0</UAS>
              </currentStatics>
              <totalStatics>
                <LCV>47</LCV>
                <PCV>6</PCV>
                <CSS>1</CSS>
                <SEFS>1</SEFS>
                <LES>1</LES>
                <DM>0</DM>
                <ES>0</ES>
                <BES>0</BES>
                <SES>0</SES>
                <UAS>2</UAS>
              </totalStatics>
            </trunkData>
          </trunkList>
        </route>
      </routeList>
    </ResponseXsvcRouteStats>
  </SOAP:Body>
</SOAP:Envelope>
```

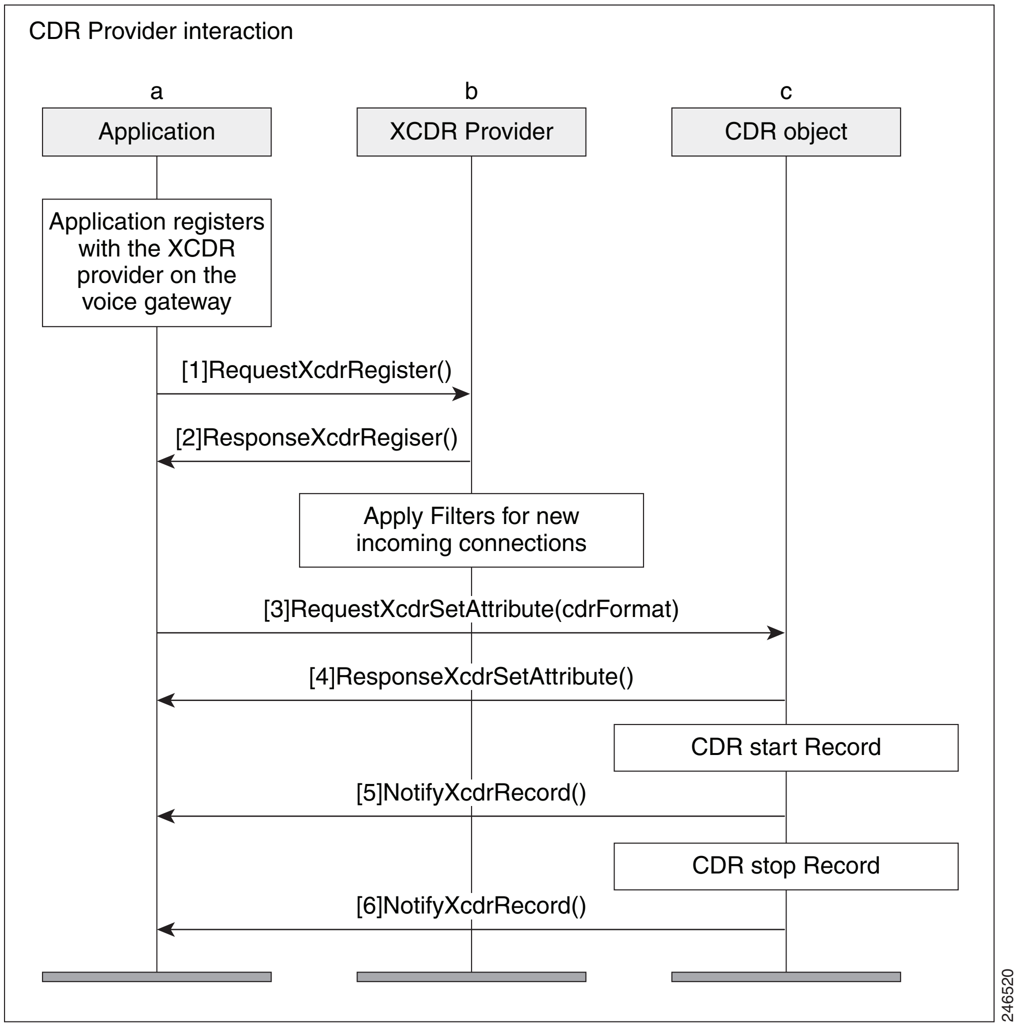
XCDR

This section describes some of the interactions that takes place etween the XCDR provider and the application.

Interaction Between the XCDR Provider and Application

Figure A-9 shows the interaction and the sequence of messages that are exchanged between the application and the XCDR provider during registration.

Figure A-9 Messae interaction when the application registers with the XCDR provider



Message Examples

This section provides examples of message exchanges between the application and the XCDR provider.

Example of a Registration Message Exchange

The following is an example of a RequestXcdrRegister message sent from the application requesting registration and specifying the type of records that it expects to receive.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <RequestXcdrRegister xmlns="http://www.cisco.com/schema/cisco_xcdr/v1_0">
      <applicationData>
        <name>myapp</name>
        <url>http://test.com:8090/xcdr</url>
      </applicationData>
      <msgHeader>
        <transactionID>txID001</transactionID>
      </msgHeader>
      <providerData>
        <url>http://10.1.1.1:8090/cisco_xcdr</url>
      </providerData>
    </RequestXcdrRegister>
  </soapenv:Body>
</soapenv:Envelope>
```

The following is an example of a ResponseXcdrRegister message sent from the XCDR provider in response to the application's registration request.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Body>
    <ResponseXcdrRegister xmlns="http://www.cisco.com/schema/cisco_xcdr/v1_0">
      <msgHeader>
        <transactionID>txID001</transactionID>
        <registrationID>152E0204:XCDR:myapp:5</registrationID>
      </msgHeader>
      <providerStatus>IN_SERVICE</providerStatus>
    </ResponseXcdrRegister>
  </SOAP:Body>
</SOAP:Envelope>
```

XMF

Message Examples

The following is an example an example of RequestXmfConnectionMediaForking with recording tone enabled.

```
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
     <RequestXmfConnectionMediaForking xmlns="http://www.cisco.com/schema/cisco_xmf/v1_0">
         <action>
            <enableMediaForking>
                 <farEndAddr>
                     <ipv4>10.104.105.232</ipv4>
                     <port>33200</port>
                     <recordTone>COUNTRY_US</recordTone>
                  </farEndAddr>
                  <nearEndAddr>
                    <ipv4>10.104.105.232</ipv4>
                    <port>45488</port>
                     <recordTone>COUNTRY_SPAIN</recordTone>
                   </nearEndAddr>
              </enableMediaForking>
          </action>
          <callID>223</callID>
          <connID>988</connID>
          <msgHeader>
             <registrationID>1CB27F48:XMF:myapp:39</registrationID>
             <transactionID>1CB2FC58:2689</transactionID>
          </msgHeader>
      </RequestXmfConnectionMediaForking>
   </soapenv:Body>
</soapenv:Envelope
```

The following is an example an example of RequestXmfCallMediaForking with recording tone enabled.

```
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
     <RequestXmfCallMediaForking xmlns="http://www.cisco.com/schema/cisco_xmf/v1_0">
       <action>
         <enableMediaForking>
            <farEndAddr>
               <ipv4>10.104.105.232</ipv4>
               <port>45388</port>
               <recordTone>COUNTRY_SPAIN</recordTone>
            </farEndAddr>
            <nearEndAddr>
              <ipv4>10.104.105.232</ipv4>
              <port>45487</port>
              <recordTone>COUNTRY_SWITZERLAND</recordTone>
            </nearEndAddr>
         </enableMediaForking>
       </action>
       <callID>305</callID>
       <msgHeader>
          <registrationID>1D50BB78:XMF:myapp:45</registrationID>
          <transactionID>1D51A394:3473</transactionID>
       </msgHeader>
    </RequestXmfCallMediaForking>
</soapenv:Body>
```

The following is an example an example of RequestXmfCallMediaSetAttributes with recording tone enabled.

```
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <RequestXmfCallMediaSetAttributes xmlns="http://www.cisco.com/schema/cisco_xmf/v1_0">
       <callID>307</callID>
       <mediaForking>
           <farEndAddr>
               <ipv4>10.104.105.232</ipv4>
               <port>32102</port>
               <recordTone>COUNTRY_GERMANY</recordTone>
           </farEndAddr>
           <nearEndAddr>
               <ipv4>10.104.105.232</ipv4>
               <port>32100</port>
               <recordTone>COUNTRY_AUSTRALIA</recordTone>
           </nearEndAddr>
       </mediaForking>
       <msgHeader>
               <registrationID>1D560670:XMF:myapp:47</registrationID>
               <transactionID>1D5690A0:3491</transactionID>
       </msgHeader>
    </RequestXmfCallMediaSetAttributes>
  </soapenv:Body>
</soapenv:Envelope>
```

Provider and Field Descriptions

XCC

XCC Provider Operations

The XCC (Extended Call Control) provider supports operations that allow a client application to perform call control and real-time call monitoring.

Provide Operation	Direction	Incoming Message	Outgoing Message	Description
XccRegister	inOut	RequestXccRegister	ResponseXccRegister fault:XMLParserError fault:ServiceException	Allows application to register with XCC provider and specify the connection events filter
XccUnRegister	inOut	RequestXccUnRegister	ResponseXmfUnRegister fault:XMLParserError fault:ServiceException	Allows application to unregister with XCC provider
XccControlUpdate	inOut	RequestXccControlUpdate	ResponseXccControlUpdate fault:XMLParserError fault:ServiceException	Allows application to update parameters after registered
XccCallRelease	inOut	RequestXccCallRelease	ResponseXccCallRelease fault:XMLParserError fault:ServiceException	Allows application to release the call session
XccConnectionRelease	inOut	RequestXccConnectionRelease	ResponseXccConnectionRelease fault:XMLParserError fault:ServiceException	Allows application to release the connection from the call session
XccProviderUnregister	outIn	ResponseXccProviderUnRegister	SolicitXccProviderUnRegister	Allows XCC Provider to unregister with application

Provide Operation	Direction	Incoming Message	Outgoing Message	Description
XccProviderStatus	OutOnly		NotifyXccProviderStatus	Updated application once XCC provider
XccCallMediaSetAttributes	inOut	RequestXccCallMediaSetAttributes	ResponseXccCallMediaSetAttributes	Allows application to specify the media attributes for a call session
XccCallMediaForking	inOut	RequestXccCallMediaForking	ResponseXccCallMediaForking fault:XMLParserError fault:ServiceException	Allows application to enable media forking a call session
XccCallData	outOnly		NotifyXccCallData	Notifies application that a call session on one of the following conditions: • mode is changed • a dtmf digit is detected • media inactive or active is detected
XccConnectionAuthorize	outIn	ResponseXccConnectionAuthorize	SolicitXccConnectionAuthorize	Allows application to perform the connection authorization
XccConnectionAuthorizeDone	inOut	RequestXccConnectionAuthorizeDone	ResponseXccConnectionAuthorizeDone fault:XMLParserError fault:ServiceException	Allows application to handle the connection once the authorization is done
XccConnectionAddressAnalyze	outIn	ResponseXccConnectionAddressAnalyze	SolicitXccConnectionAddressAnalyze	Allows application to analyze the connection address
XccConnectionAddressAnalyzeDonr	inOut	RequestXccConnectionAddressAnalyzeDone	ResponseXccConnectionAddressAnalyzeDone fault:XMLParserError fault:ServiceException	Allows application to handle the connection once the analysis is done
XccConnectionMediaForking	inOut	RequestXccConnectionMediaForking	ResponseXccConnectionMediaForking fault:XMLParserError fault:ServiceException	Allows application to enable media forking for the call session

Provide Operation	Direction	Incoming Message	Outgoing Message	Description
XccConnectionData	outOnly		NotifyXccConnectionData	Notifies application that a connection is in one of the following conditions: • a new connection is created • a connection is in call delivery state • a connection is redirected to another destination • a connection is in alerting state • a conection is in connected state • a connection is transferred to another target • a connection is in disconnected state • a connection is handoff and leave the call session • a connection is handoff to the call session
XccProbing	outIn	ResponseXccProbing	SolicitXccProbing	Allows XCC provider to keep alive a registration session and probe its health

XCC API Messages

NotifyXccCallData

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callData	CallData	M	Call information
mediaEvent	cMediaEvent	M	Choice of media event

NotifyXccConnectionData

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callData	CallData	M	Call information
connData	ConnData	M	Connection information
event	cConnectionData	M	Event choice

NotifyXccProviderStatus

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
applicationData	ApplicationData	M	Application URL configured in the router CLI
providerData	ProviderData	M	Provider data
providerStatus	eProviderStatus	M	Provider current status

RequestXccCallMediaForking

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callID	string	M	Call Identification
action	cCallMediaForking	M	Provider data

RequestXccCallMediaSetAttributes

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callID	string	M	Call identification
mediaEventsFilter	MediaEventsFilter	O	Enables media event types to be sent in an application. Turn off any media events if this element is not included in the request
mediaForking	MediaForkingData	O	Media Forking Data

RequestXccCallRelease

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callID	string	M	Call identification
disCause	int	O	Q.850 disconnect cause range [1-188]

RequestXccConnectionAddressAnalyzeDone

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callID	string	M	Call Identification
connID	string	M	Connection Identification
action	cConnectionAddressAnalyzeDone	M	Action choice

RequestXccConnectionAuthorizeDone

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callID	string	M	Call Identification
connID	string	M	Connection Identification
action	cConnectionAuthorizeDone	M	Action choice

RequestXccConnectionMediaForking

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callID	string	M	Call Identification
connID	string	M	Connection Identification
action	cCallMediaForking	M	Media forking action choice

RequestXccConnectionRelease

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callID	string	M	Call Identification
connID	string	M	Connection Identification
discCause	int	M	Q.850 disconnect cause range [1 - 188]

RequestXccControlUpdate

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
connectionEventsFilter	ConnectionEventsFilter	O	List of events that shall be notified to application
mediaEventsFilter	MediaEventsFilter	O	List of media events that shall be notfied to application
blockingEventTimeoutSec	int	O	Some application responses may block. This timeout specifies how long XCC provider will wait for the response in seconds.
blockingTimeoutHandle	eBlockingTimeoutHandle	O	How XCC provider should handle the call when blocking event timeouts

RequestXccRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
applicationData	ApplicationData	M	Application sends this request
providerData	ProviderData	M	XCC provider
connectionEventsFilter	ConnectionEventsFilter	O	List of events that shall be notified to application
mediaEventsFilter	MediaEventsFilter	O	List of media events that shall be notfied to application
blockingEventTimeoutSec	int	O	Some application responses may block. This timeout specifies how long XCC provider will wait for the response in seconds.
blockingTimeoutHandle	eBlockingTimeoutHandle	O	How XCC provider should handle the call when blocking event timeouts

RequestXccUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXccCallMediaForking

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXccCallMediaSetAttributes

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXccCallRelease

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXccConnectionAddressAnalyze

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
action	cConnectionAddressAnalyze	M	Action choice

ResponseXccConnectionAddressAnalyzeDone

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXccConnectionAuthorize

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
action	cConnectionAuthorize	M	Action choice

ResponseXccConnectionAuthorizeDone

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXccConnectionMediaForking

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXccConnectionRelease

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXccControlUpdate

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXccProbing

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
sequence	int	M	Sequence number of the probing messages

ResponseXccProviderUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXccRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
providerStatus	eProviderStatus	M	Current provider status

ResponseXccUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header the messages

SolicitXccConnectionAddressAnalyze

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callData	CallData	M	Call information
connData	ConnData	M	Connection information
collectAddress	AddrData	O	Connection collect address data

SolicitXccConnectionAuthorize

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callData	CallData	M	Call information
connDetailData	ConnDetailData	M	Connection detail information

SolicitXccProbing

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

sequence	int	M	Sequence number of the probing message
interval	duration	M	Interval between probing messages
failureCount	int	M	Counts on previous probing failures since last successful message exchange in this reigstration session
registered	boolean	M	Registration status
providerStatus	eProviderStatus	M	Provider current status

SolicitXccProviderUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

Xcc Message Data Types

This section describes the data types and elements that are found in the Xcc Provider messages.

Xcc Composite Data Type

The following section describes the composite data structures defined within the Xcc Provider.

AddrData

Referenced by: CallRouteData , ConnDetailData , RedirectAddrData, SolicitXccConnectionAddressAnalyze

Element Name	Element Type	M/O	Description
type	eAddrType	M	Address data type
addr	string	M	Address in string format

Alerting

(This is an empty element)

Block

Element Name	Element Type	M/ O	Description
blockingEventTimeoutSec	int	O	Some application responses may block. This timeout specifies how long XCC provider will wait for the response in seconds.
blockingTimeoutHandle	eBlockingTimeoutHandle	O	How XCC provider should handle the call when blocking event timeouts

CallData

Referenced by: NotifyXccCallData, NotifyXccConnectionData, SolicitXccConnectionAddressAnalyze, SolicitXccConnectionAuthorize

Element Name	Element Type	M/ O	Description
callID	string	M	Call Identification
state	eCallState	M	call state

CallDelivery

(This is an empty element)

CallRouteData

Element Name	Element Type	M/ O	Description
routeAddrData	AddrData	M	terminating party address data
connectionEventsFilter	ConnectionEventsFilter	O	List of connection events shall be enabled for the new terminating connection

cCallMediaForking

Referenced by: RequestXccCallMediaForking, RequestXccConnectionMediaForking

Element Name	Element Type	M/ O	Description
CallMediaForkingOpt	CallMediaForkingOpt - choice	M	CallMediaForkingOpt

cConnectionAddressAnalyze

Referenced by: ResponseXccConnectionAddressAnalyze

Element Name	Element Type	M/O	Description
ConnAddrAnalzOpt	ConnAddrAnalzOpt - choice	M	ConnAddrAnalzOpt

cConnectionAddressAnalyzeDone

Referenced by: RequestXccConnectionAddressAnalyzeDone

Element Name	Element Type	M/O	Description
ConnAddrAnalzDoneOpt	ConnAddrAnalzDoneOpt - choice	M	ConnAddrAnalzDoneOpt

cConnectionAuthorize

Referenced by: ResponseXccConnectionAuthorize

Element Name	Element Type	M/O	Description
ConnAuthOpt	ConnAuthOpt - choice	M	ConnAuthOpt

cConnectionAuthorizeDone

Referenced by: RequestXccConnectionAuthorizeDone

Element Name	Element Type	M/O	Description
ConnAuthDoneOpt	ConnAuthDoneOpt - choice	M	ConnAuthDoneOpt

cConnectionData

Referenced by: NotifyXccConnectionData

Element Name	Element Type	M/O	Description
ConnDataOpt	ConnDataOpt - choice	M	ConnDataOpt

cMediaEvent

Referenced by: NotifyXccCallData

Element Name	Element Type	M/O	Description
MediaEventOpt	MediaEventOpt - choice	M	MediaEventOpt

ConnData

Referenced by: ConnDetailData , NotifyXccConnectionData, SolicitXccConnectionAddressAnalyze

Element Name	Element Type	M/O	Description
connID	string	M	Connection Identification
state	eConnState	M	connection state

ConnDetailData

Referenced by: Connected , Created, HandoffJoin , SolicitXccConnectionAuthorize

Element Name	Element Type	M/O	Description
connData	ConnData	M	Connection information
guid	string	M	Connection guid data
guidAltFormat	string	O	Connection guid data represented in Alternate format
callingAddrData	AddrData	O	Calling party address data
origCallingAddrData	AddrData	O	orignal calling party address data
calledAddrData	AddrData	O	Called party address data

origCalledAddrData	AddrData	O	original called party address data
redirectAddrData	RedirectAddrData	O	Redirect party address data
connIntfType	eConnIntfType	O	Connection interface type
mediaData	MediaData	O	Connection media data
connIntf	string	O	Connection interface name string
connDirectionType	eConnDirectionType	M	Connection direction type
routeName	string	O	Connection interface route name string
routeDescription	string	O	Route description

Connected

Element Name	Element Type	M/O	Description
connDetailData	ConnDetailData	M	Connection detail information

ConnectionEventsFilter

Referenced by: CallRouteData , RequestXccControlUpdate, RequestXccRegister

Element Name	Element Type	M/O	Description
eConnectionEventsFilter	eMediaEventsFilter	O	

ContinueProcessing

(This is an empty element)

Created

Element Name	Element Type	M/O	Description
connDetailData	ConnDetailData	M	Connection detail information

DisableMediaForking

(This is an empty element)

Disconnected

Element Name	Element Type	M/O	Description
mediaData	MediaData	M	Connection media data
discCause	int	M	Q.850 disconnect cause range [1 - 188]
statsData	StatsData	O	statistics data
jitterData	JitterData	O	media jitter data

DTMF

Element Name	Element Type	M/O	Description
digit	string	M	a dtmf digit
dateTime	string	M	Time when dtmf occurs

HandoffJoin

Element Name	Element Type	M/O	Description
connDetailData	ConnDetailData	M	Connection detail information

HandoffLeave

(This is an empty element)

JitterData

Referenced by: Disconnected

Element Name	Element Type	M/O	Description
roundTripDelayMSec	int	M	Round trip delay (in ms)
onTimeRvPlayMSec	int	M	On time Rv Play (in ms)
gapFillWithPredictionMSec	int	M	Prediction count (in ms)
gapFillWithInterpolationMSec	int	M	Interpolation count (in ms)
gapFillWithRedundancyMSec	int	M	Redundancy count (in ms)
lostPacketsCount	int	M	Lost packets count
earlyPacketsCount	int	M	Early packets count

latePacketsCount	int	M	Late packets count
receiveDelayMSec	int	M	Receive delay (in ms)
loWaterPlayoutDelayMSec	int	M	Low water playout delay (in ms)
hiWaterPlayoutDelayMSec	int	M	Hi water playout delay (in ms)

MediaActivity

Element Name	Element Type	M/O	Description
old	eActivityState	M	old media activity state
new	eActivityState	M	new media activity state

MediaAddrData

Referenced by: MediaForkingData

Element Name	Element Type	M/O	Description
ipv4	string	M	Remote IP Address ver 4
port	int	M	Remote RTP port
recordTone	eCountryType	O	Country specific record tone

MediaData

Referenced by: ConnDetailData , Disconnected

Element Name	Element Type	M/O	Description
type	eMediaType	M	Media type
coderType	string	O	codec type
coderByte	int	O	codec byte

MediaEventsFilter

Referenced by: RequestXccCallMediaSetAttributes, RequestXccControlUpdate, RequestXccControlUpdate

Element Name	Element Type	M/O	Description
eMediaEventsFilter	MediaEventsFilter	O	

MediaForkingData

Referenced by:RequestXccCallMediaSetAttributes

Element Name	Element Type	M/O	Description
nearEndAddr	MediaAddrData	M	Media address for near-end side
farEndAddr	MediaAddrData	M	Media address for far-end side
preserve	boolean	O	Media Forking Preservd after app unregister

MediaForkingEvent

Element Name	Element Type	M/O	Description
mediaForkingState	eMediaForkingState	M	Media forking status

ModeChange

Element Name	Element Type	M/O	Description
old	eMediaType	M	old media type
new	eMediaType	M	new media type

RedirectAddrData

Referenced by: ConnDetailData , Redirected, Transferred

Element Name	Element Type	M/O	Description
calledAddrData	AddrData	M	called address data

Redirected

Element Name	Element Type	M/O	Description
redirectAddrData	RedirectAddrData	M	Redirect party address data

Release

Element Name	Element Type	M/O	Description
discCause	int	M	Q.850 disconnect cause range [1 - 188]

StatsData

Referenced by: Disconnected

Element Name	Element Type	M/O	Description
callDuration	duration	M	call duration
TxPacketsCount	int	M	Total Tx Packets
TxBytesCount	int	M	Total Tx Bytes
TxDurationMSec	int	M	Tx Duration in milliseconds
TxVoiceDurationMSec	int	M	Tx Voice Duration in milliseconds
RxPacketsCount	int	M	Total Rx Packets
RxBytesCount	int	M	Total Rx Bytes
RxDurationMSec	int	M	Rx Duration in milliseconds
RxVoiceDurationMSec	int	M	Rx Voice Duration in milliseconds

Tone

Element Name	Element Type	M/O	Description
toneType	eToneType	M	Tone type

Transferred

Element Name	Element Type	M/O	Description
redirectAddrData	RedirectAddrData	O	Redirect party address data

Xcc Choice Elements

Choice records - may contain only one field at a time

CallMediaForkingOpt - choice

Referenced by: cCallMediaForking

Enable media forking Only one of the following elements:

Element Name	Element Type	Desciption
enableMediaForking	MediaForkingData	Enable media forking
disableMediaForking	Empty element	Disable media forking

ConnAddrAnalzDoneOpt - choice

Referenced by: cConnectionAddressAnalyzeDone

Release the connection Only one of the following elements:

Element Name	Element Type	Desciption
release	Release	Release the connection
continueProcessing	Empty element	Continue the connection processing
callRoute	CallRouteData	Application specifies the call route

ConnAddrAnalzOpt - choice

Referenced by: cConnectionAddressAnalyze

Temporary block the connection processing and wait for application for further request Only one of the following elements:

Element Name	Element Type	Desciption
block	Block	Temporary block the connection processing and wait for application for further request
release	Release	Release the connection
continueProcessing	Empty element	Continue the connection processing
callRoute	CallRouteData	Application specifies the call route

ConnAuthDoneOpt - choice

Referenced by: cConnectionAuthorizeDone

Release the connection Only one of the following elements:

Element Name	Element Type	Desciption
release	Release	Release the connection
continueProcessing	Empty element	Continue the connection processing

ConnAuthOpt - choice

Referenced by: cConnectionAuthorize

Temporary block the connection processing and wait for application for further request Only one of the following elements:

Element Name	Element Type	Desciption
block	Block	Temporary block the connection processing and wait for application for further request
release	Release	Release the connection
continueProcessing	Empty element	Continue the connection processing

ConnDataOpt - choice

Referenced by: cConnectionData

Enables connection created notify event Only one of the following elements:

Element Name	Element Type	Desciption
created	Created	Enables connection created notify event
callDelivery	Empty element	Enables call delivery notify event
alerting	Empty element	Enables connection alerting notify event

redirected	Redirected	Enables connection redirected notify event
connected	Connected	Enables connection connected notify event
transferred	Transferred	Enables connection transferred notify event
disconnected	Disconnected	Enables connection disconnected notify event
handoffLeave	Empty element	Enables connection handoff leave notify event
handoffJoin	HandoffJoin	Enables connection handoff join notify event
mediaForking	MediaForkingEvent	Updates media forking status

MediaEventOpt - choice

Referenced by: cMediaEvent

DTMF detected Only one of the following elements:

Element Name	Element Type	Desciption
DTMF	DTMF	DTMF detected
mediaActivity	MediaActivity	Media activity state changed
modeChange	ModeChange	Mode of call changed
tone	Tone	Tone detected
mediaForking	MediaForkingEvent	Updates media forking status

Xcc Enumerated Elements

This section describes the enumerated elements that are found in the Xcc provider data types and Xcc provider messages.

eActivityState

Referenced by: MediaActivity

Value	Description
ACTIVE	Active state
INACTIVE	Inactive state

eAddrType

Referenced by: AddrData

Value	Description
E164	Address is e164 number format

URI	Address is URI string format
OTHER	Address in other formats

eBlockingTimeoutHandle

Referenced by: Block , RequestXccControlUpdate, RequestXccControlUpdate

Value	Description
RELEASE	Abort connection attempt
CONTINUE_PROCESSING	Proceed with connection attempt

eCallState

Referenced by: CallData

Value	Description
IDLE	Initial state of a call. A call has zero connection
ACTIVE	A call has ongoing activity
INVALID	Final state of a call. A call in this state has one or more connections associated with

eConnDirectionType

Referenced by: ConnDetailData

Value	Description
INCOMING	Incoming connection
OUTGOING	Outgoing connection

eConnectionEventsFilter

Referenced by: ConnectionEventsFilter

Value	Description
CREATED	First event sent when a new connection is created
AUTHORIZE_CALL	Sent to request call authorization
ADDRESS_ANALYZE	Enables address analyze solicit event
REDIRECTED	Enables connection redirected notify event

ALERTING	Enables connection alerting notify event
CONNECTED	Enables connection connected notify event
TRANSFERRED	Enables connection transferred notify event
CALL_DELIVERY	Enables connection call delivery notify event
DISCONNECTED	Enables connection disconnected notify event
HANDOFFLEAVE	Enables connection handoff leave notify event
HANDOFFJOIN	Enables connection handoff join notify event

eConnIntfType

Referenced by: ConnDetailData

Value	Description
CONN_UNKNOWN	Unknown connection interface type
CONN_ANALOG_EM	Analog E n M port
CONN_ANALOG_FXO	Analog FXO port
CONN_ANALOG_FXS	Analog FXS port
CONN_ANALOG_EFXS	Analog eFXS port
CONN_ANALOG_EFXO	Analog eFXO port
CONN_ISDN	ISDN PRI interface
CONN_CAS	CAS interfacee
CONN_BRI	ISDN BRI interface
CONN_R2	E1 R2 interface
CONN_H323	H.323 interface
CONN_SIP	SIP interface
CONN_TRUNKGROUP	Trunk group

eConnState

Referenced by: ConnData

Value	Description
IDLE	Connection is idle state
AUTHORIZE_CALL_ATTEMPT	Connection is in authorize call attempt
ADDRESS_COLLECT	Connection is in collecting address state
ADDRESS_ANALYZE	Connection is pending for address analyze state
CALL_DELIVERY	Connection is in call delivery state

ALERTING	Connection is in alerting state
CONNECTED	Connection is in connected state
DISCONNECTED	Enables connection disconnected notify event

eCountryType

Referenced by: MediaAddrData

Value	Description
COUNTRY_USA	United States
COUNTRY_AUSTRALIA	Australia
COUNTRY_GERMANY	Germany
COUNTRY_RUSSIA	Russia
COUNTRY_SPAIN	Spain
COUNTRY_SWITZERLAND	Switzerland

eMediaEventsFilter

Referenced by: MediaEventsFilterMediaEventsFilter

Value	Description
DTMF	Enables inband dtmf detection
MEDIA_ACTIVITY	Enables media activity detection
MODE_CHANGE	Enables mode change notify when a mode of a call session has changed
TONE_BUSY	Enables busy tone detection
TONE_DIAL	Enables dialtone detection
TONE_OUT_OF_SERVICE	Enable out of service tone detection
TONE_RINGBACK	Enables ringback detection
TONE_SECOND_DIAL	Enables secondary dialtone detection

eMediaForkingState

Referenced by: MediaForkingEvent

Value	Description
FORK_STARTED	Media forking setup success

FORK_FAILED	Media forking setup failure
FORK_DONE	Media forking completed

eMediaType

Referenced by: MediaData, ModeChange

Value	**Description**
VOICE	Voice call
FAX	Fax call
MODEM	Modem call
VIDEO	Video call
DATA	Data call

eToneType

Referenced by: Tone

Value	**Description**
TONE_BUSY	busy tone detected
TONE_DIAL	dialtone detected
TONE_RINGBACK	ringback detected
TONE_SECOND_DIAL	secondary dialtone detected
TONE_OUT_OF_SERVICE	out of service detected

XSVC

Xsvc Provider Operations

The XSVC provider monitors the trunk status, and provides real-time notification of link status and configuration change to application.

Provide Operation	Direction	Incoming Message	Outgoing Message	Description
XsvcRegister	inOut	RequestXsvcRegister	ResponseXsvcRegister fault: XMLParserError fault: ServiceException	Allows application to register with XSVC provider and specify the connection events filter
XsvcUnRegister	inOut	RequestXsvcUnRegister	ResponseXsvcUnRegister fault: XMLParserError fault: ServiceException	Allows application to unregister with XSVC provider
XsvcProviderUnRegister	outIn	ResponseXsvcProviderUnRegister	SolicitXsvcProviderUnRegister	Allows XSVC provider to unregister with application.
XsvcProviderStatus	outOnly		NotifyXsvcProviderStatus	Updates application once the XSVC provider status has changed
XsvcRouteSetFilter	inOut	RequestXsvcRouteSetFilter	ResponseXsvcRouteSetFilter fault: XMLParserError fault: ServiceException	Allows the application to set the fitler so that XSVC provider will only report to the application the updates it is interested in
XsvcRouteSnapshot	inOut	RequestXsvcRouteSnapshot	ResponseXsvcRouteSnapshot fault: XMLParserError fault: ServiceException	Allows application to get the big picture of all the routes being monitored.
XsvcRouteStats	inOut	RequestXsvcRouteStats	ResponseXsvcRouteStats fault: XMLParserError fault: ServiceException	Allows application to query the statistics of a trunk

XsvcRouteData	inOut	RequestXsvcRouteData	ResponseXsvcRouteData fault: XMLParserError fault: ServiceException	Allows application to query the detail information of a trunk
XsvcRouteConfiguration	outOnly		NotifyXsvcRouteConfiguration	Notifies application that a trunk configuration is changed
XsvcRouteStatus	outOnly		NotifyXsvcRouteStatus	Notifies application that a trunk status is change: Link status is changed Alarm status is changed
XsvcProbing	outIn	ResponseXsvcProbing	SolicitXsvcProbing	Allows XSVC provider to keep alive a registration session and probe its health.

Xsvc API Messages

NotifyXsvcProviderStatus

Element Name	Element Type	M/ O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
applicationData	ApplicationData	M	Application URL configured in router CLI
providerData	ProviderData	M	Provider data
providerStatus	eProviderStatus	M	Provider current status

NotifyXsvcRouteConfiguration

Element Name	Element Type	M/ O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
type	eRouteChangeType	M	
routeList	RouteList	M	Compact form of route information

NotifyXsvcRouteStatus

Element Name	Element Type	M/ O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
routeList	RouteList	M	Compact form of route information

RequestXsvcRegister

Element Name	Element Type	M/ O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
applicationData	ApplicationData	M	Application sends this request
providerData	ProviderData	M	XSVC provider
routeEventsFilter	RouteEventsFilter	O	List of events that shall be notified to application

RequestXsvcRouteData

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
routeName	string	M	Route name
routeType	eRouteType	M	Route type

RequestXsvcRouteSetFilter

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
isOn	boolean	M	
routeFilterList	RouteFilterList	O	Route filter list

RequestXsvcRouteSnapshot

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

RequestXsvcRouteStats

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
routeName	string	M	Route name
routeType	eRouteType	M	Route type

RequestXsvcUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXsvcProbing

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
sequence	int	M	Sequence number of the probing messages

ResponseXsvcProviderUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXsvcRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
providerStatus	eProviderStatus	M	Current providerstatus

ResponseXsvcRouteData

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
routeList	RouteList	M	Compact form of route information

ResponseXsvcRouteSetFilter

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXsvcRouteSnapshot

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
routeList	RouteList	M	Compact form of route information

ResponseXsvcRouteStats

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
routeList	RouteList	M	Compact form of route information

ResponseXsvcUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

SolicitXsvcProbing

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
sequence	int	M	Sequence number of the probing message
interval	duration	M	Interval between probing messages
failureCount	int	M	Counts on previous probing failures since last successful message exchange in this reigstration session
registered	boolean	M	Registration status
providerStatus	eProviderStatus	M	Provider current status

SolicitXsvcProviderUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

Xsvc Message Data Types

This section describes the data types and elements that are found in the Xsvc Provider messages.

Xsvc Composite Data Type

The following section describes the composite data structures defined within the Xsvc Provider.

CurrentStatistics

Referenced by: IntfStatisticsData

Element Name	Element Type	M/O	Description
elapsedTime	duration	M	The time have elapsed since the beginning of the far end current error-measurement period
LCV	int	M	Line Coding Violation Error Event
PCV	int	M	Path Coding Violation Error Event
CSS	int	M	Controlled Slip Seconds
SEFS	int	M	Severely Errored Framing Second
LES	int	M	Line Errored Seconds
DM	int	M	Degraded Minutes
ES	int	M	Errored Seconds
BES	int	M	Bursty Errored Seconds
SES	int	M	everely Errored Seconds
UAS	int	M	Unavailable Seconds

IntfChannels

Referenced by: TrunkData

Element Name	Element Type	M/O	Description

channels	string	O	Channel mapping of the interface
totalChannels	int	M	Total channels on the interface

IntfStatisticsData

Referenced by: TrunkData

Element Name	Element Type	M/O	Description
currentStatistics	CurrentStatistics	M	Interface latest statistics
totalStatistics	TotalStatistics	M	Interface accumulated statistics

RouteData

Referenced by: RouteList

Element Name	Element Type	M/O	Description
routeName	string	M	Route name
routeType	eRouteType	M	Route type
routeDescription	string	O	Route description
trunkList	TrunkList	O	

RouteEventsFilter

Referenced by: RequestXsvcRegister

Element Name	Element Type	M/O	Description
eRouteEventsFilter	eRouteEventsFilter	O	

RouteFilter

Referenced by: RouteFilterList

Element Name	Element Type	M/O	Description
routeName	string	M	Route name
routeType	eRouteType	M	Route type

RouteFilterList

Referenced by: RequestXsvcRouteSetFilter

Element Name	Element Type	M/O	Description
routeFilter	RouteFilter	O	Route filter

RouteList

Referenced by: NotifyXsvcRouteConfiguration, NotifyXsvcRouteStatus, ResponseXsvcRouteData, ResponseXsvcRouteSnapshot, ResponseXsvcRouteStats

Element Name	Element Type	M/O	Description
route	RouteData	M	

TotalStatistics

Referenced by: IntfStatisticsData

Element Name	Element Type	M/O	Description
intervalTime	duration	M	The time of previous far end intervals for which data was collected
LCV	int	M	Line Coding Violation Error Event
PCV	int	M	Path Coding Violation Error Event
CSS	int	M	Controlled Slip Seconds
SEFS	int	M	Severely Errored Framing Second
LES	int	M	Line Errored Seconds
DM	int	M	Degraded Minutes
ES	int	M	Errored Seconds
BES	int	M	Bursty Errored Seconds
SES	int	M	everely Errored Seconds
UAS	int	M	Unavailable Seconds

TrunkData

Referenced by: TrunkList

List of one or more connection events

Element Name	Element Type	M/ O	Description
name	string	M	Name of the turnk interface
type	eTrunkType	M	Type of the turnk interface
status	eTrunkStatus	M	Status of the turnk interface
channelData	IntfChannels	O	Trunk interface channel information
alarmData	eTrunkAlarm	O	Trunk interface alarm information
statisticsData	IntfStatisticsData	O	Trunk interface statistics information

TrunkList

Referenced by: RouteData

Element Name	Element Type	M/ O	Description
trunkData	TrunkData	M	

Xsvc Enumerated Elements

This section describes the enumerated elements that are found in the Xsvc provider data types and Xsvc provider messages.

eRouteChangeType

Referenced by: NotifyXsvcRouteConfiguration

Value	Description
ROUTE_ADDED	
ROUTE_DELETED	
ROUTE_MODIFIED	

eRouteEventsFilter

Referenced by: RouteEventsFilter

Value	Description
ROUTE_CONF_UPDATED	Enables route configuration updated notify event
ROUTE_STATUS_UPDATED	Enables route status updated notify event)

eRouteType

Referenced by: RequestXsvcRouteData, RequestXsvcRouteStats, RouteData, RouteFilter

Value	Description
VOIP	
PSTN	

eTrunkAlarm

Referenced by: TrunkData

Value	Description
NoAlarm	No alarm present
RcvFarEndLOF	Far end LOF (a.k.a. Yellow Alarm)
XmtFarEndLOF	Near end sending LOF Indication
RcvAIS	Far end sending AIS
XmtAIS	Near end sending AIS
LossOfFrame	Near end LOF (a.k.a. Red Alarm)
LossOfSignal	Near end loss Of Signal
LoopbackState	Near end is looped
T16AIS	E1 TS16 AIS
RcvFarEndOLMF	Far End Send TS16 LOMF
XmtFarEndOLMF	Near End Send TS16 LOMF
RcvTestCode	Near End detects a test code
OtherFailure	any line status not defined here
UnavailSigState	Near End in Unavailable Signal State
NetEquipOOS	Carrier Equipment Our Of Service
RcvPayloadAIS	DS2 Payload AIS
Ds2PerfThreshold	DS2 Performance Threshold

eTrunkStatus

Referenced by: TrunkData

Value	Description
UP	-
DOWN	-

eTrunkType

Referenced by: TrunkData

Value	Description
ISDN_PRI	-
ISDN_BRI	-
ANALOG	-
CAS	-
SIPV2	-
H323	-

XCDR

Xcdr Provider Operations

The XCDR provider provides CDR information for the application. It notifies the application when calls are set up or ended.

Provide Operation	Direction	Incoming Message	Outgoing Message	Description
XcdrRegister	inOut	RequestXcdrRegister	ResponseXcdrRegister fault: XMLParserError fault: ServiceException	Allows application to register with XCDR provider and specify the connection events filter
XcdrUnRegister	inOut	RequestXcdrUnRegister	ResponseXcdrUnRegister fault: XMLParserError fault: ServiceException	Allows application to unregister with XCDR provider
XcdrProviderUnRegister	outIn	ResponseXcdrProviderUnRegister	SolicitXcdrProviderUnRegister	Allows XCDR provider to unregister with application.
XcdrProviderStatus	outOnly		NotifyXcdrProviderStatus	Updates application once the XCDR provider status has changed

XcdrSetAttribute	inOut	RequestXcdrSetAttribute	ResponseXcdrSetAttribute	Allows application to specify the attribute it is needed. Two formats, compact or detailed, can be selected.
XcdrRecord	outOnly		NotifyXcdrRecord	Notifies application the CDR
XcdrProbing	outIn	ResponseXcdrProbing	SolicitXcdrProbing	Allows XCDR provider to keep alive a registration session and probe its health.

Xcdr API Messages

NotifyXcdrProviderStatus

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
applicationData	ApplicationData	M	Application URL configured in router CLI
providerData	ProviderData	M	Provider data
providerStatus	eProviderStatus	M	Provider current status

NotifyXcdrRecord

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
format	eCdrFormat	M	CDR format
type	eCdrType	M	CDR type
cdr	string	M	CDR information

RequestXcdrRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
applicationData	ApplicationData	M	Application send s this request
providerData	ProviderData	M	XCDR provider
cdrEventsFilter	CdrEventsFilter	O	List of events that shall be notified to application

RequestXcdrSetAttribute

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
format	eCdrFormat	M	CDR format

RequestXcdrUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXcdrProbing

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
sequence	int	M	Sequence number of the probing messages

ResponseXcdrProviderUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXcdrRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
providerStatus	eProviderStatus	M	Current providerstatus

ResponseXcdrSetAttribute

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXcdrUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

SolicitXcdrProbing

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
sequence	int	M	Sequence number of the probing message
interval	duration	M	Interval between probing messages
failureCount	int	M	Counts on previous probing failures since last successful message exchange in this registration session
registered	boolean	M	Registration status
providerStatus	eProviderStatus	M	Provider current status

SolicitXcdrProviderUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

Xcdr Message Data Types

This section describes the data types and elements that are found in the Xcdr Provider messages.

Xcdr Composite Data Type

The following section describes the composite data structures defined within the Xcdr Provider.

CdrEventsFilter

Referenced by: RequestXcdrRegister

list of one or more CDR events

Element Name	Element Type	M/O	Description
eCdrEventsFilter	eCdrEventsFilter	O	

Xcdr Enumerated Elements

This section describes the enumerated elements that are found in the Xcdr provider data types and Xcdr provider messages.

eCdrEventsFilter

Referenced by: CdrEventsFilter

Value	Description
CDR_RECORD	Enables CDR record notify event

eCdrFormat

Referenced by: NotifyXcdrRecord, RequestXcdrSetAttribute

Value	Description
COMPACT	Displaying CDR in compact format
DETAIL	Displaying CDR in detail format

eCdrType

Referenced by: NotifyXcdrRecord

Value	Description
START	CDR when call are set up
STOP	CDR when call are released

XMF

Xmf Provider Operations

The XMF (Extended Media Forking) provider supports operations that allow a client application to perform media forking and real-time call monitoring.

Provide Operation	Direction	Incoming Message	Outgoing Message	Description
XmfRegister	inOut	RequestXmfRegister	ResponseXmfRegister fault: XMLParserError fault: ServiceException	Allows application to register with XMF provider and specify the connection events filter
XmfUnRegister	inOut	RequestXmfUnRegister	ResponseXmfUnRegister fault: XMLParserError fault: ServiceException	Allows application to unregister with XMF provider
XmfControlUpdate	inOut	RequestXmfControlUpdate	ResponseXmfControlUpdate fault: XMLParserError fault: ServiceException	Allows application to update parameters after registered
XmfProviderUnRegister	outIn	ResponseXmfProviderUnRegister	SolicitXmfProviderUnRegister	Allows XMF provider to unregister with application
XmfProviderStatus	outOnly		NotifyXmfProviderStatus	Updates application once the XMF provider status has changed
XmfCallMediaSetAttributes	inOut	RequestXmfCallMediaSetAttributes	ResponseXmfCallMediaSetAttributes fault: XMLParserError fault: ServiceException	Allows application to specify the media attributes for the call session

XmfCallMediaForking	inOut	RequestXmfCallMediaForking	ResponseXmfCallMediaForking fault: XMLParserError fault: ServiceException	Allows application to enable media forking for the call session
XmfConnectionMediaForking	inOut	RequestXmfConnectionMediaForking	ResponseXmfConnectionMediaForking fault: XMLParserError fault: ServiceException	Allows application to enable media forking for the connection
XmfCallData	outOnly		NotifyXmfCallData	Notifies application that a call session on one of the following conditions: • mode is changed • a dtmf digit is detected • media inactive or active is detected

XmfConnection Data	outOnly		NotifyXmfConnectionData	Notifies application that a connection is in one of the following conditions: a new connection is created a connection is in call delivery state a connection is redirected to another destination a connection is in alerting state a conection is in connected state a connection is transferred to another target a connection is in disconnected state a connection is handoff and leave the call session a connection is handoff to the call session
XmfProbing	outIn	ResponseXmfProbing	SolicitXmfProbing	Allows XMF provider to keep alive a registration session and probe its health

Xmf API Messages

NotifyXmfCallData

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callData	CallData	M	call information
mediaEvent	cMediaEvent	M	Choice of media event

NotifyXmfConnectionData

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callData	CallData	M	Call information
connData	ConnData	M	Connection information
event	cConnectionData	M	Event choice

NotifyXmfProviderStatus

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
applicationData	ApplicationData	M	Application URL configured in router CLI
providerData	ProviderData	M	Provider data
providerStatus	eProviderStatus	M	Provider current status

RequestXmfCallMediaForking

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callID	string	M	Call Identification
action	cCallMediaForking	M	Media forking action choice

RequestXmfCallMediaSetAttributes

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callID	string	M	Call Identification
mediaEventsFilter	MediaEventsFilter	O	Enables media event types to be sent to application. Turn off any media events if this element is not included in the request
mediaForking	MediaForkingData	O	Media forkig data

RequestXmfConnectionMediaForking

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
callID	string	M	Call Identification
connID	string	M	Connection Identification
action	cCallMediaForking	M	Media forking action choice

RequestXmfControlUpdate

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
connectionEventsFilter	ConnectionEventsFilter	O	List of events that shall be notified to application
mediaEventsFilter	MediaEventsFilter	O	List of media events that shall be notfied to application

RequestXmfRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
applicationData	ApplicationData	M	Application sends this request
providerData	ProviderData	M	XMF provider

connectionEventsFilter	ConnectionEventsFilter	O	List of events that shall be notified to application
mediaEventsFilter	MediaEventsFilter	O	List of media events that shall be notfied to application

RequestXmfUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXmfCallMediaForking

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXmfCallMediaSetAttributes

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXmfConnectionMediaForking

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXmfControlUpdate

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXmfProbing

Element Name	Element Type	M/O	Description

msgHeader	MsgHeader	M	Message header common for all the messages
sequence	int	M	Sequence number of the probing messages

ResponseXmfProviderUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

ResponseXmfRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
providerStatus	eProviderStatus	M	Current provider status

ResponseXmfUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header the messages

SolicitXmfProbing

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages
sequence	int	M	Sequence number of the probing message
interval	duration	M	Interval between probing messages
failureCount	int	M	Counts on previous probing failures since last successful message exchange in this reigstration session
registered	boolean	M	Registration status
providerStatus	eProviderStatus	M	Provider current status

SolicitXmfProviderUnRegister

Element Name	Element Type	M/O	Description
msgHeader	MsgHeader	M	Message header common for all the messages

Xmf Message Data Types

This section describes the data types and elements that are found in the Xmf Provider messages.

Xmf Composite Data Type

The following section describes the composite data structures defined within the Xmf Provider.

AddrData

Referenced by: ConnDetailData , RedirectAddrData

Element Name	Element Type	M/O	Description
type	eAddrType	M	Address data type
addr	string	M	Address in string format

Alerting

(This is an empty element)

CallData

Referenced by: NotifyXmfCallData, NotifyXmfConnectionData

Element Name	Element Type	M/O	Description
callID	string	M	Call Identification
state	eCallState	M	call state

CallDelivery

(This is an empty element)

cCallMediaForking

Referenced by: RequestXmfCallMediaForking, RequestXmfConnectionMediaForking

Element Name	Element Type	M/O	Description
CallMediaForkingOpt	CallMediaForkingOpt - choice	M	CallMediaForkingOpt

cConnectionData

Referenced by: NotifyXmfConnectionData

Element Name	Element Type	M/O	Description
ConnDataOpt	ConnDataOpt - choice	M	ConnDataOpt

cMediaEvent

Referenced by: NotifyXmfCallData

Element Name	Element Type	M/O	Description
MediaEventOpt	MediaEventOpt - choice	M	MediaEventOpt

ConnData

Referenced by: ConnDetailData , NotifyXmfConnectionData

Element Name	Element Type	M/O	Description
connID	string	M	Connection Identification
state	eConnState	M	connection state

ConnDetailData

Referenced by: Connected , Created, HandoffJoin

Element Name	Element Type	M/O	Description
connData	ConnData	M	Connection information

guid	string	M	Connection guid data
guidAltFormat	string	O	Connection guid data represented in Alternate format
callingAddrData	AddrData	O	Calling party address data
origCallingAddrData	AddrData	O	orignal calling party address data
calledAddrData	AddrData	O	Called party address data
origCalledAddrData	AddrData	O	original called party address data
redirectAddrData	RedirectAddrData	O	Redirect party address data
connIntfType	eConnIntfType	O	Connection interface type
mediaData	MediaData	O	Connection media data
connIntf	string	O	Connection interface name string
connDirectionType	eConnDirectionType	M	Connection direction type
routeName	string	O	Connection interface route name string
routeDescription	string	O	Route description

Connected

Element Name	Element Type	M/O	Description
connDetailData	ConnDetailData	M	Connection detail information

ConnectionEventsFilter

Referenced by: RequestXmfControlUpdate, RequestXmfRegister

Element Name	Element Type	M/O	Description
eConnectionEventsFilter	eConnectionEventsFilter	O	

Created

Element Name	Element Type	M/O	Description
connDetailData	ConnDetailData	M	Connection detail information

DisableMediaForking

(This is an empty element)

Disconnected

Element Name	Element Type	M/O	Description
mediaData	MediaData	M	Connection media data
discCause	int	M	Q.850 disconnect cause range [1 - 188]
statsData	StatsData	O	statistics data
jitterData	JitterData	O	media jitter data

DTMF

Element Name	Element Type	M/O	Description
digit	string	M	a dtmf digit
dateTime	string	M	Time when dtmf occurs

HandoffJoin

Element Name	Element Type	M/O	Description
connDetailData	ConnDetailData	M	Connection detail information

HandoffLeave

(This is an empty element)

JitterData

Referenced by: Disconnected

Element Name	Element Type	M/O	Description
roundTripDelayMSec	int	M	Round trip delay (in ms)
onTimeRvPlayMSec	int	M	On time Rv Play (in ms)
gapFillWithPredictionMSec	int	M	Prediction count (in ms)
gapFillWithInterpolationMSec	int	M	Interpolation count (in ms)
gapFillWithRedundancyMSec	int	M	Redundancy count (in ms)
lostPacketsCount	int	M	Lost packets count
earlyPacketsCount	int	M	Early packets count
latePacketsCount	int	M	Late packets count
receiveDelayMSec	int	M	Receive delay (in ms)
loWaterPlayoutDelayMSec	int	M	Low water playout delay (in ms)
hiWaterPlayoutDelayMSec	int	M	Hi water playout delay (in ms)

MediaActivity

Element Name	Element Type	M/O	Description
old	eActivityState	M	old media activity state
new	eActivityState	M	new media activity state

MediaAddrData

Referenced by: MediaForkingData

Element Name	Element Type	M/O	Description
ipv4	string	M	Remote IP Address ver 4
port	int	M	Remote RTP port
recordTone	eCountryType	O	Country specific record Tone

MediaData

Referenced by: ConnDetailData , Disconnected

Element Name	Element Type	M/O	Description
type	eMediaType	M	Media type
coderType	string	O	codec type
coderByte	int	O	codec byte

MediaEventsFilter

Referenced by: RequestXmfCallMediaSetAttributes, RequestXmfControlUpdate, RequestXmfRegister

Element Name	Element Type	M/O	Description
eMediaEventsFilter	eMediaEventsFilter	O	

MediaForkingData

Referenced by: RequestXmfCallMediaSetAttributes

Element Name	Element Type	M/O	Description
nearEndAddr	MediaAddrData	M	Media address for near-end side
farEndAddr	MediaAddrData	M	Media address for far-end side
preserve	boolean	O	Media Forking Preservd after app unregister

MediaForkingEvent

Element Name	Element Type	M/O	Description
mediaForkingState	eMediaForkingState	M	Media forking status

ModeChange

Element Name	Element Type	M/O	Description
old	eMediaType	M	old media type
new	eMediaType	M	new media type

RedirectAddrData

Referenced by: ConnDetailData , Redirected, Transferred

Element Name	Element Type	M/O	Description
calledAddrData	AddrData	M	called address data

Redirected

Element Name	Element Type	M/O	Description
redirectAddrData	RedirectAddrData	M	Redirect party address data

StatsData

Referenced by: Disconnected

Element Name	Element Type	M/O	Description
callDuration	duration	M	call duration
TxPacketsCount	int	M	Total Tx Packets
TxBytesCount	int	M	Total Tx Bytes
TxDurationMSec	int	M	Tx Duration in milliseconds

TxVoiceDurationMSec	int	M	Tx Voice Duration in milliseconds
RxPacketsCount	int	M	Total Rx Packets
RxBytesCount	int	M	Total Rx Bytes
RxDurationMSec	int	M	Rx Duration in milliseconds
RxVoiceDurationMSec	int	M	Rx Voice Duration in milliseconds

Tone

Element Name	Element Type	M/O	Description
toneType	eToneType	M	Tone type

Transferred

Element Name	Element Type	M/O	Description
redirectAddrData	RedirectAddrData	O	Redirect party address data

Xmf Choice Elements

Choice records - may contain only one field at a time

CallMediaForkingOpt - choice

Referenced by: cCallMediaForking

Enable media forking Only one of the following elements:

Element Name	Element Type	Description
enableMediaForking	MediaForkingData	Enable media forking
disableMediaForking	Empty element	Disable media forking

ConnDataOpt - choice

Referenced by: cConnectionData

Enables connection created notify event Only one of the following elements:

Element Name	Element Type	Description
created	Created	Enables connection created notify event
callDelivery	Empty element	Enables call delivery notify event
alerting	Empty element	Enables connection alerting notify event
redirected	Redirected	Enables connection redirected notify event
connected	Connected	Enables connection connected notify event
transferred	Transferred	Enables connection transferred notify event
disconnected	Disconnected	Enables connection disconnected notify event
handoffLeave	Empty element	Enables connection handoff leave notify event
handoffJoin	HandoffJoin	Enables connection handoff join notify event
mediaForking	MediaForkingEvent	Updates media forking status

MediaEventOpt - choice

Referenced by: cMediaEvent

DTMF detected Only one of the following elements:

Element Name	Element Type	Description
DTMF	DTMF	DTMF detected
mediaActivity	MediaActivity	Media activity state changed
modeChange	ModeChange	Mode of call changed
tone	Tone	Tone detected
mediaForking	MediaForkingData	Updates media forking status

Xmf Enumerated Elements

This section describes the enumerated elements that are found in the Xmf provider data types and Xmf provider messages.

eActivityState

Referenced by: MediaActivity

Value	Description
ACTIVE	Active state
INACTIVE	Inactive state

eAddrType

Referenced by: AddrData

Value	Description
E164	Address is e164 number format
URI	Address is URI string format
OTHER	Address in other formats

eCallState

Referenced by: CallData

Value	Description
IDLE	Initial state of a call. A call has zero connection
ACTIVE	A call has ongoing activity
INVALID	Final state of a call. A call in this state has one or more connections associated with

eConnDirectionType

Referenced by: ConnDetailData

Value	Description
INCOMING	Incoming connection
OUTGOING	Outgoing connection

eConnectionEventsFilter

Referenced by: ConnectionEventsFilter

Value	Description
CREATED	First event sent when a new connection is created
REDIRECTED	Enables connection redirected notify event
ALERTING	Enables connection alerting notify event
CONNECTED	Enables connection connected notify event
TRANSFERRED	Enables connection transferred notify event
CALL_DELIVERY	Enables connection call delivery notify event
DISCONNECTED	Enables connection disconnected notify event

HANDOFFLEAVE	Enables connection handoff leave notify event
HANDOFFJOIN	Enables connection handoff join notify event

eConnIntfType

Referenced by: ConnDetailData

Value	Description
CONN_UNKNOWN	Unknown connection interface type
CONN_ANALOG_EM	Analog E n M port
CONN_ANALOG_FXO	Analog FXO port
CONN_ANALOG_FXS	Analog FXS port
CONN_ANALOG_EFXS	Analog eFXS port
CONN_ANALOG_EFXO	Analog eFXO port
CONN_ISDN	ISDN PRI interface
CONN_CAS	CAS interfacee
CONN_BRI	ISDN BRI interface
CONN_R2	E1 R2 interface
CONN_H323	H.323 interface
CONN_SIP	SIP interface
CONN_TRUNKGROUP	Trunk group

eConnState

Referenced by: ConnData

Value	Description
IDLE	Connection is idle state
AUTHORIZE_CALL_ATTEMPT	Connection is in authorize call attempt
ADDRESS_COLLECT	Connection is in collecting address state
ADDRESS_ANALYZE	Connection is pending for address analyze state
CALL_DELIVERY	Connection is in call delivery state
ALERTING	Connection is in alerting state
CONNECTED	Connection is in connected state
DISCONNECTED	Enables connection disconnected notify event

eCountryType

Referenced by: MediaAddrData

Value	Description
COUNTRY_USA	United States
COUNTRY_AUSTRALIA	Australia
COUNTRY_GERMANY	Germany
COUNTRY_RUSSIA	Russia
COUNTRY_SPAIN	Spain
COUNTRY_SWITZERLAND	Switzerland

eMediaEventsFilter

Referenced by: MediaEventsFilter

Value	Description
DTMF	Enables inband dtmf detection
MEDIA_ACTIVITY	Enables media activity detection
MODE_CHANGE	Enables mode change notify when a mode of a call session has changed
TONE_BUSY	Enables busy tone detection
TONE_DIAL	Enables dialtone detection
TONE_OUT_OF_SERVICE	Enable out of service tone detection
TONE_RINGBACK	Enables ringback detection
TONE_SECOND_DIAL	Enables secondary dialtone detection

eMediaForkingState

Referenced by: MediaForkingEvent

Value	Description
FORK_STARTED	Media forking setup success
FORK_FAILED	Media forking setup failure
FORK_DONE	Media forking completed

eMediaType

Referenced by: MediaData, ModeChange

Value	Description
VOICE	Voice call
FAX	Fax call
MODEM	Modem call
VIDEO	Video call
DATA	Data call

eToneType

Referenced by: Tone

Value	Description
TONE_BUSY	busy tone detected
TONE_DIAL	dialtone detected
TONE_RINGBACK	ringback detected
TONE_SECOND_DIAL	secondary dialtone detected
TONE_OUT_OF_SERVICE	out of service detected

Common Message Data Types

This section describes the data types and elements that are found in the Common Module messages.

Common Composite Data Type

The following section describes the composite data structures defined within the Common Module.

ApplicationData

Element Name	Element Type	M/O	Description
url	anyURI	M	Application url data
name	string	O	Application name

MsgHeader

Element Name	Element Type	M/O	Description
transactionID	string	O	ID to identify a transaction for the message excahnge between provider and application. This filed is optional. This field is mandatory for the response message to return the same transactionID if present in the request/solicit message.
registrationID	string	O	ID to identify a registration session. This field is absent for RequestRegister and NotifyStatus messages. This field is mandatory for all the other messages.

ProviderData

Element Name	Element Type	M/O	Description
url	anyURI	M	url for client application

Common Enumerated Elements

This section describes the enumerated elements that are found in the Common Module data types.

eProviderStatus

Value	Description
SHUTDOWN	Service is not running
IN_SERVICE	Service is enabled and running

Common Module

Common Message Data Types

This section describes the data types and elements that are found in the Common Module messages.

Common Composite Data Type

The following section describes the composite data structures defined within the Common Module.

ApplicationData

Element Name	Element Type	M/ O	Description
url	anyURI	M	Application url data
name	string	O	Application name

MsgHeader

Element Name	Element Type	M/ O	Description
transactionID	string	O	ID to identify a transaction for the message excahnge between provider and application. This filed is optional. This field is mandatory for the response message to return the same transactionID if present in the request/solicit message.
registrationID	string	O	ID to identify a registration session. This field is absent for RequestRegister and NotifyStatus messages. This field is mandatory for all the other messages.

ProviderData

Element Name	Element Type	M/ O	Description
url	anyURI	M	url for client application

Common Enumerated Elements

This section describes the enumerated elements that are found in the Common Module data types.

eProviderStatus

Value	Description
SHUTDOWN	Service is not running
IN_SERVICE	Service is enabled and running

Common XML Types

The following types are defined by XML:

Type	Reference
any	http://www.w3.org/TR/xmlschema-2/#any
anyURI	http://www.w3.org/TR/xmlschema-2/#anyURI
boolean	http://www.w3.org/TR/xmlschema-2/#boolean
dateTime	http://www.w3.org/TR/xmlschema-2/#dateTime
duration	http://www.w3.org/TR/xmlschema-2/#duration
int	http://www.w3.org/TR/xmlschema-2/#int
name	http://www.w3.org/TR/xmlschema-2/#Name
string	http://www.w3.org/TR/xmlschema-2/#string

Fault Module

Fault Message Data Types

This section describes the data types and elements that are found in the Fault Module messages.

Fault Composite Data Type

The following section describes the composite data structures defined within the Fault Module.

ServiceException

The service exception fault bound to SOAP fault elements are listed:

Element Name	Element Type	M/ O	Description
Soap:Code/Value	string	M	The value is "Receiver"
Soap:Code/Subcode /Value	string	O	The value is "SERVICE EXCEPTION"

Soap:Reason/Text	string	M	Information on the nature of the fault
Soap:Detail	ServiceException	O	Details of the service exception.

The elemenet ServiceException is defined as:

Element Name	Element Type	M/ O	Description
errorCode	string	M	Error identifier with service prefix and number. Refer to ServiceExceptionErrorCode for detail.
operation	string	O	Service opertion of the message
transactionID	string	O	transactionID if present in the request.
registrationID	string	O	registrationID if present in the request.
text	string	O	Message text

XMLParserError

When the SOAP message contains syntax error, the XML parser will fail, and a SOAP fault message will be generated. The XML parser error fault bound to SOAP fault elements are listed:

Element Name	Element Type	M/ O	Description
Soap:Code/Value	string	M	The value is "Sender" or "Receiver"
Soap:Code/Subcode /Value	string	M	The value is "XML PARSER ERROR"

Soap:Reason/Text	string	M	Information on the nature of the fault as follows: Memory exhausted Badly framed XML received Unknown namespace received A required attribute is missing An uninterpretable attribute has been received An invalidattribute value has been received An unknown XML tag has been received Anexpected XML tag or sequence is missing An unexpected XML tag has been received The value for an XML tag is not valid An internal error caused processing to be aborted An unsupported operation request has been received
Soap:Detail	XMLParserError	O	Details of the XML parser error.

The elemenet XMLParserError is defined as:

Element Name	Element Type	M/O	Description
errorXMLDetail	string	O	Information to identify where is the parsing error in the XML message
errorXMLMsg	any	O	A copy of the original XML message for debugging purpose.
errorXMLTag	string	O	XML tag which causes the failure

Fault XML Types

The following types are defined by XML:

Type	Reference
any	http://www.w3.org/TR/xmlschema-2/#any
anyURI	http://www.w3.org/TR/xmlschema-2/#anyURI
boolean	http://www.w3.org/TR/xmlschema-2/#boolean
dateTime	http://www.w3.org/TR/xmlschema-2/#dateTime
duration	http://www.w3.org/TR/xmlschema-2/#duration
int	http://www.w3.org/TR/xmlschema-2/#int

name	http://www.w3.org/TR/xmlschema-2/#Name
string	http://www.w3.org/TR/xmlschema-2/#string