Cisco Unified Communications Manager Developers Guide

Release 6.0(1)

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: OL-12410-01

CONTENTS

Preface

This section explains the objectives, intended audience, and organization of this publication and describes the conventions that convey instructions and other information.

This preface includes the following sections:

Purpose

This document describes the following Cisco Unified Communications Manager (formerly Cisco Unified CallManager) APIs:

- The Cisco Unified Communications Manager AXL implementation allows applications to modify the Cisco Unified Communications Manager system database. Be aware that AXL is not intended as a real-time API but as a provisioning and configuration API.
- Cisco Unified Communications Manager real-time information, performance counters, and database information exposure occur through the AXL Serviceability API.
- The Cisco Unified Communications Manager Extension Mobility Service provides a rich API, which enables extension mobility on IP phones and allows application control over authentication, scheduling, and availability. It allows a device, usually a Cisco Unified IP Phone, to temporarily embody a new device profile, including lines, speed dials, and services. An application that uses the Cisco Unified Communications Manager Extension Mobility Service represents an IP phone service that allows a user to log in by entering a userID and PIN. The architecture and implementation of the Cisco Unified Communications Manager Extension Mobility Service make many other applications possible.

  Some examples follow:

  - An application that automatically activates phones for employees when they reserve a particular desk for a particular time (the scheduling application).
  - A lobby phone does not have a line appearance until a user logs in.
- The Cisco Unified Communications Manager Web Dialer application, which is installed on a Cisco Unified Communications Manager server, enables click-to-dial functionality by creating hyperlinked telephone numbers in a company directory. This functionality allows users to make calls from a web page by clicking the telephone number of the person that they are trying to call. The Web Dialer application, which has a SOAP interface, uses JavaScript to provide the web page functionality.

Audience

The *Cisco Unified Communications Manager Developers Guide* provides information for developers who write applications that extend the functionality of the APIs that are described in this document.

This guide assumes the developer has knowledge of a high-level programming language such as C++, Java, or an equivalent language. You must also have knowledge or experience in the following areas:

- Extensible Markup Language (XML)
- Hypertext Markup Language (HTML)
- Hypertext Transport Protocol (HTTP)
- Simple Object Access Protocol (SOAP) 1.1
- Socket programming
- TCP/IP Protocol
- Web Service Definition Language (WSDL) 1.1
- Secure Sockets Layer (SSL)

In addition, users of the Cisco Unified Communications Manager APIs must have a firm grasp of XML Schema. For more information on XML Schema, refer to http://www.w3.org/TR/xmlschema-0/.

The developer must also have an understanding of Cisco Unified Communications Manager and its applications. The Related Documentation section lists documents on Cisco Unified Communications Manager and other related technologies.

Organization

The following organization applies for this guide.

Table 1 Organization

Chapter	Description
Chapter 1, "AXL Configuration Programming"	This chapter describes the Administrative XML Layer (AXL) API, which provides a mechanism for inserting, retrieving, updating, and removing data from the database by using an XML SOAP interface. This API lets you access Cisco Unified Communications Manager data by using XML and receive the data in XML form.
Chapter 2, "AXL Serviceability API Programming"	This chapter describes the AXL Serviceability APIs, which are based on Java Servlets on the Apache Tomcat web server. Cisco Unified Communications Manager real-time information, performance counters, and database information exposure occurs through the AXL Serviceability APIs.
Chapter 3, "Cisco Extension Mobility Service API"	This chapter includes high-level concepts that are important in understanding the Cisco Extension Mobility Service as well as an overview of configuring EM services, messages, message DTDs, and error codes.
Chapter 4, "Cisco Web Dialer API Programming"	This chapter describes the Simple Object Access Protocol (SOAP) and HTML over HTTP (and HTTPS) interfaces that are used to develop JavaScript-based directory search web pages and applications for Cisco Web Dialer.

New and Changed Information

The *Cisco Unified Communications Manager New and Changed Information Guide for Release 6.0(1)* describes new features and or changes that are pertinent to release 6.0 of the Cisco Unified Communications Manager.

See also New and Changed Information in Chapter 1, "AXL Configuration Programming."

Related Documentation

This section lists documents and URLs that provide information on Cisco Unified Communications Manager, Cisco Unified IP Phones, and the technologies that are required to develop applications.

- Cisco Unified Communications Manager Release 6.0—A suite of documents that relate to the installation and configuration of Cisco Unified Communications Manager. Refer to the *Cisco Unified Communications Manager Documentation Guide for Release 6.0* for a list of documents on installing and configuring Cisco Unified Communications Manager 6.0, including

  - *Cisco Unified Communications Manager Administration Guide, Release 6.0.*
  - *Cisco Unified Communications Manager System Guide, Release 6.0.*
  - *Cisco Unified Communications Manager Features and Services Guide, Release 6.0.*
- *Cisco Unified IP Phones and Services*—A suite of documents that relate to the installation and configuration of Cisco Unified IP Phones.
- *Cisco DistributedDirector*—A suite of documents that relate to the installation and configuration of Cisco DistributedDirector.

Related Information

- Simple Object Access Protocol (SOAP) 1.1
- Web Service Definition Language (WSDL) 1.1
- SOAP Tutorial
- WSDL Tutorial—Web Service Definition Language tutorial.
- http://www.soapagent.com/—Open SOAP directory with links to articles, tutorials, and white papers.

Conventions

This document uses the following conventions:

Convention	Description
boldface font	Commands and keywords are in **boldface**.
italic font	Arguments for which you supply values are in *italics*.
[ ]	Elements in square brackets are optional.
{ x | y | z }	Alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A non-quoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
`screen` font	Terminal sessions and information the system displays are in `screen` font.
`boldface screen` font	Information you must enter is in **`boldface screen`** font.
`italic screen` font	Arguments for which you supply values are in *`italic screen`* font.
→	This pointer highlights an important line of text in an example.

Convention	Description
^	The symbol ^ represents the key labeled Control—for example, the key combination ^D in a screen display means hold down the Control key while you press the D key.
< >	Non-printing characters, such as passwords, are in angle brackets.

Notes use the following conventions:

Note Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the publication.

Timesavers use the following conventions:

Timesaver Means *the described action saves time*. You can save time by performing the action described in the paragraph.

Tips use the following conventions:

Tip Means *the following are useful tips.*

Cautions use the following conventions:

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

Warnings use the following conventions:

Warning **This warning symbol means danger. You are in a situation that could cause bodily injury. Before you work on any equipment, you must be aware of the hazards involved with electrical circuitry and familiar with standard practices for preventing accidents.**

Obtaining Documentation, Obtaining Support, and Security Guidelines

For information on obtaining documentation, obtaining support, providing documentation feedback, security guidelines, and also recommended aliases and general Cisco documents, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Cisco Product Security Overview

This product contains cryptographic features and is subject to United States and local country laws governing import, export, transfer and use. Delivery of Cisco cryptographic products does not imply third-party authority to import, export, distribute or use encryption. Importers, exporters, distributors and users are responsible for compliance with U.S. and local country laws. By using this product you agree to comply with applicable laws and regulations. If you are unable to comply with U.S. and local laws, return this product immediately.

A summary of U.S. laws governing Cisco cryptographic products may be found at: http://www.cisco.com/wwl/export/crypto/tool/stqrg.html. If you require further assistance please contact us by sending email to export@cisco.com.

CHAPTER 1

AXL Configuration Programming

The Administrative XML Layer (AXL) Application Programming Interface (API) provides a mechanism for inserting, retrieving, updating, and removing data from the Cisco Unified Communications Manager database by using an eXtensible Markup Language (XML) Simple Object Access Protocol (SOAP) interface. This allows a programmer to access the database by using XML and receive the data in XML form, instead of using a binary library or DLL.

The AXL API methods, known as requests, use a combination of HTTPS and SOAP. SOAP is an XML remote procedure call (RPC) protocol. Users perform requests by sending XML data to the Cisco Unified Communications Manager server. The server then returns the AXL response, which is also a SOAP message.

The AXL-SOAP web service is enabled by default on all Cisco Unified Communications Manager servers. It requires no other installation or configuration.

To access all AXL SOAP API downloads and AXL requests and responses that are found in this chapter, refer to http://www.cisco.com/pcgi-bin/dev_support/access_level/product_support.

This chapter assumes the developer has knowledge of a high-level programming language such as C++, Java, or an equivalent language.

Developers must also have knowledge or experience in the following areas:

- TCP/IP Protocol
- Hypertext Transport Protocol (specifically HTTPS)
- Socket programming
- XML

Tip Users of the AXL API must have a firm grasp of XML syntax and Schema, which is used to define the AXL requests, responses, and errors.

For more information on XML Schema, refer to http://www.w3.org/TR/xmlschema-0/.

For more information on XML syntax/grammar, refer to http://www.w3.org/TR/rdf-syntax-grammar/.

Caution The AXL API allows you to modify the Cisco Unified Communications Manager system database. You must use caution when using AXL, because each API call impacts the system. Misuse of the API can lead to dropped calls and slower performance. AXL should act as a provisioning and configuration API, not as a real-time API.

This chapter contains the following sections:

New and Changed Information

Release 6.0(1) makes the following major changes in the AXL APIs for :

- Introduced AXL schema versioning (see AXL Versioning Support).
- Added the following three fields into the AXL device/line create APIs:
  - ASCII Display (internal call ID)
  - Secondary Calling Search Space for Forward All
  - Unattended Port
- Added new AXL APIs and attributes that are needed to support Mobility provisioning and added optional attributes to existing AXL objects.
- Added the new tag cfaCSSPolicy in the Line API.
- Added Credential Policy support APIs.
- Added information about AXL versioning.
- Changed the default value of the AXL service parameter, so it allows a valid namespace to be returned in AXL responses.
- In Cisco Unified Communications Manager Release 6.0(1), the AXL API now considers PKID with or without curly brackets as a valid input. For example, in an addLine request for the DevicePool tag, PKID can be mentioned either as “{xxx-xxx...}” or “xxx-xxx...”; both represent acceptable inputs for the AXL request.

AXL Versioning Support

To improve backward compatibility, Cisco Unified Communications Manager Release 6.0(1) introduces AXL schema versioning. The system duplicates the previous AXL 1.0 schema as the AXL 6.0(1) schema, and, in upcoming releases, the AXL schema will be numbered the same as the corresponding Cisco Unified Communications Manager release. This approach maintains AXL backward compatibility for one full release cycle.

Developers have the option in Cisco Unified Communications Manager Release 6.0(1) to request a specific AXL schema version (either 1.0 or 6.0(1)); however, both these schema versions are identical. If a specific schema version is not requested, the AXL 1.0/6.0(1) schema automatically gets used and an appropriate response gets returned, which will have no backward compatibility issues for several releases.

Cisco highly recommends that developers include the version of AXL schema on which an AXL request is based because support for unversioned requests might be removed in future releases of Cisco Unified Communications Manager.

For those developers who are using the AXL APIs executeSQLQuery and updateSQLquery, changes have occurred to the Cisco Unified Communications Manager 6.0(1) database schema that affect the direct SQL query approach. Refer to the Cisco Unified Communications Manager *Database Dictionary, Release 6.0(1),* which describes the specific changes in the database schema.

To help developers plan for AXL versioning, Table 1-1 provides the approach that Cisco will follow in supporting upcoming releases.

- Cisco will support AXL requests ***without*** version information for only three releases following the 6.0(1) release; after that, requests without version information might be rejected.
- AXL requests ***with*** version information will have the corresponding schema applied for up to three subsequent releases; after that, the specified version might not be available.

Table 1-1 AXL Versioning and Schema Plan for Future Releases

Cisco Unified Communications Manager Release	AXL Request no version specified	AXL Request with Version Specified				
		Release 6.0	Plus 1 release	Plus 2 releases	Plus 3 releases	Plus 4 releases
Release 6.0	6.0 schema applied	6.0 schema applied				
Plus 1 release	6.0 schema applied	6.0 schema applied	Plus 1 schema applied	Not Applicable		
Plus 2 releases	6.0 schema applied	6.0 schema applied	Plus 1 schema applied	Plus 2 schema applied		
Plus 3 releases	6.0 schema applied	6.0 schema applied	Plus 1 schema applied	Plus 2 schema applied	Plus 3 schema applied	
Plus 4 releases	Request rejected		Plus 1 schema applied	Plus 2 schema applied	Plus 3 schema applied	Plus 4 schema applied
Plus 5 releases	Request rejected	Schema no longer available		Plus 2 schema applied	Plus 3 schema applied	Plus 4 schema applied

The following a sample AXL request carries version information:

```
POST /axl/ HTTP/1.0
Host:10.77.31.194:8443
Authorization: Basic Q0NNQWRtaW5pc3RyYXRvcjpjaXNjb19jaXNjbw==
Accept: text/*
Content-type: text/xml
SOAPAction: "CUCM:DB ver=6.0"
Content-length: 427

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <SOAP-ENV:Body>
<axl:getUser xmlns:axl="http://www.cisco.com/AXL/1.0"
xsi:schemaLocation="http://www.cisco.com/AXL/1.0 http://ccmserver/schema/axlsoap.xsd"
sequence="1234"> <userid>tttt</userid> </axl:getUser>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sample AXL Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONIDSSO=950805DE5E10F32C5788AE164EEC4955; Path=/
Set-Cookie: JSESSIONID=151CF94ACF20728B1D47CC5C3BECC401; Path=/axl; Secure
SOAPAction: "CUCM:DB ver=6.0"
Content-Type: text/xml;charset=utf-8
Content-Length: 728
Date: Mon, 22 Jan 2007 06:51:42 GMT
Connection: close
```

AXL APIs Added for Cisco Unified Communications Manager 6.0(1)

The following list provides AXL API calls that are new in Release 6.0(1):

- addAARGroup
- removeAARGroup
- updateAARGroup
- getAARGroup
- updateAARGroupMatrix
- addApplicationToSoftkeyTemplate
- removeApplicationToSoftkeyTemplate
- addCallerFilterList
- removeCallerFilterList
- updateCallerFilterList
- getCallerFilterList
- getCCMVersion
- addCommonDeviceConfig
- removeCommonDeviceConfig
- updateCommonDeviceConfig
- getCommonDeviceConfig
- addCredentialPolicy
- removeCredentialPolicy
- updateCredentialPolicy
- getCredentialPolicy
- addIVRUserLocale
- removeIVRUserLocale
- updateIVRUserLocale
- getIVRUserLocale
- updateLicenseCapabilities
- getLicenseCapabilities
- addMeetMe
- removeMeetMe
- updateMeetMe
- getMeetMe
- addMobileVoiceAccess
- removeMobileVoiceAccess
- updateMobileVoiceAccess
- getMobileVoiceAccess
- addMobility
- removeMobility
- updateMobility

- getMobility
- addMOHServer
- removeMOHServer
- updateMOHServer
- getMOHServer
- addPhoneTemplate
- removePhoneTemplate
- updatePhoneTemplate
- getPhoneTemplate
- addPhysicalLocation
- removePhysicalLocation
- updatePhysicalLocation
- getPhysicalLocation
- addRecordingProfile
- removeRecordingProfile
- updateRecordingProfile
- getRecordingProfile
- addRemoteDestination
- removeRemoteDestination
- updateRemoteDestination
- getRemoteDestination
- addRemoteDestinationProfile
- removeRemoteDestinationProfile
- updateRemoteDestinationProfile
- getRemoteDestinationProfile
- addSoftKeyTemplate
- updateSoftKeyTemplate
- getSoftKeyTemplate
- removeSoftKeyTemplate
- updateSoftKeySet
- getSoftKeySet
- addTranscoder
- updateTranscoder
- getTranscoder
- removeTranscoder
- addTransformationPattern
- removeTransformationPattern
- updateTransformationPattern
- getTransformationPattern

Note The getCCMVersion API will return the Cisco Unified Communications Manager Version based on the Node Name that is specified in the request. If no Node Name is specified, you will get the Cisco Unified Communications Manager Version of the lowest node ID Cisco Unified Communications Manager.

Cisco always advises running the AXL API on a completely upgraded cluster. When run on a cluster that is not upgraded completely, the response of the AXL API will be correct when executed on a server that already has been upgraded. However, if you execute the AXL API on a server that has not yet been upgraded, then it will return the Cisco Unified Communications Manager Version of the lowest node ID Cisco Unified Communications Manager per the server local database information.

AXL APIs Changed for Cisco Unified Communications Manager 6.0(1)

The following AXL API calls changed since the previous release. These changes might require changes to existing user code that is making use of them:

- updateAppUser
- addCallPickupGroup
- updateCallPickupGroup
- getCallPickupGroup
- addConferenceBridge
- updateConferenceBridge
- getConferenceBridge
- addCSS
- updateCSS
- getCSS
- addDevicePool
  - In axl.xsd, the tag name aarNeighborhood and the annotation for the revertPriority tag in XDevicePool changed to match the AXL response.
  - In axlsoap.xsd, the tag name aarNeighborhood and the annotation for the revertPriority tag in updateDevicePoolReq changed.
- updateDevicePool
  - In axl.xsd, the tag name aarNeighborhood and the annotation for the revertPriority tag in XDevicePool changed to match the AXL response.
  - In axlsoap.xsd, the tag name aarNeighborhood and the annotation for the revertPriority tag in updateDevicePoolReq changed.
- getDevicePool
  - In axl.xsd, the tag name aarNeighborhood and the annotation for the revertPriority tag in XDevicePool changed to match the AXL response.
  - In axlsoap.xsd, the tag name aarNeighborhood and the annotation for the revertPriority tag in updateDevicePoolReq changed.
- addDeviceProfile
- updateDeviceProfile
- getDeviceProfile

- addGatewayEndpoint
- updateGatewayEndpoint
- getGatewayEndpoint
- addH323Phone
- updateH323Phone
- getH323Phone
- addH323Trunk
- updateH323Trunk
- getH323Trunk
- addLine
- updateLine
- getLine
- addMGCP
- getMGCP
- addPhone
- updatePhone
- getPhone
- addRegion
- updateRegion
- getRegion
- updateRegionMatrix
- addRoutePartition
- updateRoutePartition
- getRoutePartition
- addSIPTrunk
- updateSIPTrunk
- getSIPTrunk
- addUser
- updateUser
- getUser
- addVoiceMailPort
- updateVoiceMailPort
- getVoiceMailPort
- doAuthenticateUser
- getCMCInfo, removeCMCInfo, and updateCMCInfo

  In axlsoap.xsd

  - A new option tag “code” along with the “uuid” tag for these three requests exists. The user can send either the uuid or code tag.

  - This release renames the existing tag “code” to “newCode” in the updateCMCInfo request. Users can send the new code to be updated as the “newCode” tag instead of “code” in update CMCInfo requests.
  - This release removes the invalid authorizationLevel tag in the updateCMCInfo request.
- addFACInfo, getFACInfo, updateFACInfo, and removeFACInfo

  In axl.xsd

  - The existing tag “description” changed to “name.” This makes the addFACInfo, getFACInfo, and updateFACInfo request match the database. In previous releases, the value that was supplied for the “description” tag updated “name” in the database.

  In axlsoap.xsd

  - A new option tag “name” along with the “uuid” tag for getFACInfo, updateFACInfo, and removeFACInfo exist.
  - The existing tag “description” changed to “newName” in the updateFACInfo request.
  - Users can send either uuid or name in the getFACInfo, updateFACInfo, and removeFACInfo requests.

This release deprecated some of the fields that were removed from the Cisco Unified Communications Manager 6.0(1) database in AXL. This release adds annotation for such fields.

Service Parameters Added or Changed

The Cisco Unified Communications Manager 6.0(1) release adds a new service parameter, EnableAXLEncodingInfo, to the Cisco Unified Communications Manager Administrator windows under the Cisco Database Layer Monitor service. This parameter allows the user to decide whether AXL responses should contain the encoding information. Consider encoding information as important if an AXL request has non-English characters in it.

The Cisco Unified Communications Manager 5.1(1) release added a new service parameter, Send Valid Namespace in the AXL response, under the Cisco Database Layer Monitor service. This parameter determines the namespace that is sent in the AXL response from the Cisco Unified Communications Manager. In the 6.0(1) release, the default value of this parameter changed from False to True.

- When this parameter is True, Cisco Unified Communications Manager sends the valid namespace (https://www.cisco.com/AXL/API/1.0) in the AXL response, so the namespace matches the AXL schema specification.
- If the parameter is False, Cisco Unified Communications Manager sends an invalid namespace (https://www.cisco.com/AXL/1.0) in the AXL response, which does not match the AXL schema specification.

  To maintain backward compatibility with older applications, you might need to change the value to False. Cisco recommends that you set this parameter to True, so the Cisco Unified Communications Manager sends a valid namespace.

AXL APIs Added for Cisco Unified Communications Manager 5.1(1)

The following list provides AXL API calls that were added in Release 5.1(1):

- addSIPRealm
- updateSIPRealm

- getSIPRealm
- removeSIPRealm

These APIs add and update credentials (passwordreserve) in siprealm.

AXL API

Request methods represent XML structures that are passed to the AXL API server. The server receives the XML structures and executes the request. If the request completes successfully, the system returns the appropriate AXL response. All responses get named identically to the associated requests, except that the word “Response” is appended.

For example, the XML response that is returned from an addPhone request is named addPhoneResponse.

If an error occurs, an XML error structure gets returned wrapped inside a SOAP Fault structure (see the “AXL Error Codes” section on page 1-10).

AXL Compliance

The Cisco Unified Communications Manager AXL implementation complies with XML Schema 1.0, which was tested for XML Schema compliance with a third-party application that is called XML Spy version 4.x. Early versions of the MSXML schema validator did not support enough of the XML Schema 1.0 recommendation to be used.

The Cisco Unified Communications Manager AXL implementation also complies with SOAP 1.1 as defined by the World Wide Web Consortium as well as HTTPS 1.1. The AXL API runs as an independent service that can be accessed only via HTTPS.

AXL Error Codes

If an exception occurs on the server, or if any other error occurs during the processing of an AXL request, the system returns an error in the form of a SOAP Fault message:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Client</faultcode>
            <faultstring>
            <![CDATA[
            An error occurred during parsing
            Message: End element was missing the character '>'.
            Source = Line : 41, Char : 6
            Code : c00ce55f, Source Text : </re
            ]]>
            </faultstring>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Fault messages can also contain more detailed information. The following example depicts a detailed SOAP Fault.

```
<SOAP-ENV:Envelope  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Client</faultcode>
            <faultstring>Device not found with name SEP003094C39708.</faultstring>
            <detail  xmlns:axl="http://www.cisco.com/AXL/1.0"
                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:schemaLocation="http://www.cisco.com/AXL/1.0
                    http://myhost/CCMApi/AXL/V1/axlsoap.xsd">
                <axl:error  sequence="1234">
                    <code>0</code>
                    <message>
<![CDATA[
Device not found with name SEP003094C39708.
]]>
                    </message>
                    <request>doDeviceLogin</request>
                </axl:error>
            </detail>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The <detail> element of a SOAP Fault includes error codes. The axl:Error elements represent the errors. If a response to a request contains an <error> element, the user agent can determine the cause of the error by looking at the subelements of the <error> tag.

The following list describes the <error> elements:ws The user agent uses the <code> element, a numerical value, to find what type of error occurred.

The error codes follow:

Error Code	Description
5000	**Unknown Error**—An unknown error occurred while the request was processed. This can occur due to a problem on the server but can also be due to errors in the request.
5002	**Unknown Request Error**—The user agent submitted a request that is unknown to the API.
5003	**Invalid Value Exception**—The API detected an invalid value in the XML request.
5007	**Item Not Valid Error**—The system identified the specified item as invalid, which means that it does not exist or that it was specified incorrectly at input.

message

The system provides the <message> element, so the user agent gets a detailed error message that explains the error.

request

The system provides the <request> element, so the user agent can determine the type of request that generated this error. Because this element is optional, it may not always appear.

Example AXL Requests

No platform considerations exist in Cisco Unified Communications Manager Release 6.0(1). The client must be able to send an HTTPS request to the AXL endpoint.

The following examples describe how to make an AXL request and read back the response to the request.

Ensure each SOAP request is sent to the web server via an HTTPS POST. The endpoint URL represents the AXL web service that is running on a Cisco Unified Communications Manager server. The following list contains the only four required HTTPS headers.

- POST :8443/axl/

  The first header specifies that this particular POST is intended for the Cisco AXL Web Service. The AXL API only responds to the POST method.

- content-type: text/xml

  The second header confirms that the data that is being sent to AXL is XML. If this header is not found, the system returns an HTTP 415 error to the client.

- Authorization: Basic <some Base 64 encoded string>

  The third header gives the Base64 encoding of the user name and password for the administrator of the AXL Server. Because Base64 encoding takes three 8-bit bytes and represents them as four printable ASCII characters, if the encoded header does not contain an even multiple of four ASCII characters (16, 20, 24, and so on), you must add padding characters (=) to complete the final group of four as in the following examples.

  If authentication of the user fails, the system returns an HTTP 401 Access Denied error to the client.

- content-length: <a positive integer>

  The fourth header specifies the length (in bytes) of the AXL request.

Note Currently, the content length cannot exceed 40 kilobytes. If a request is received that is greater than 40 kilobytes, the system returns an HTTP 413 error message.

The following example contains an HTTPS header for an AXL SOAP request:

```
POST :8443/axl/
Host: axl.myhost.com:8443
Accept: text/*
Authorization: Basic bGFycnk6Y3VybHkgYW5kIG1vZQ==
Content-type: text/xml
SOAPAction: "CUCM:DB ver=6.0"
Content-length: 613
```

The following AXL request gets used in the code examples that display in the following sections. This example shows a getPhone request:

```
POST :8443/axl/
Host: axl.myhost.com:8443
Accept: text/*
Authorization: Basic bGFycnk6Y3VybHkgYW5kIG1vZQ==
Content-type: text/xml
SOAPAction: "CUCM:DB ver=6.0"
Content-length: 613

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
        <axl:getPhone xmlns:axl="http://www.cisco.com/AXL/1.0"
xsi:schemaLocation="http://www.cisco.com/AXL/1.0 http://ccmserver/schema/axlsoap.xsd
"  sequence="1234">
            <phoneName>SEP222222222245</phoneName>
        </axl:getPhone>
    </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

C or C++ Example

This code example uses a hard-coded AXL request and sends it to the AXL server that is running on the local system (localhost). It then reads the response and outputs the response to the screen.

```
#include <sys/socket.h>
#include <sys/types.h>
#include <stdlib.h>
#include <openssl/ssl.h>
#include <stdio.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <strings.h>
#include <openssl/x509.h>
#include <openssl/crypto.h>
#include <iostream>
#include <string>
using namespace std;
typedef unsigned char byte;



void encodeBase64( const string& inBuf, string &outBuf )
{
    unsigned int i;
    unsigned int j;
    bool hiteof = false;
    byte dtable[256];

    outBuf.erase();

    for(i= 0;i<9;i++)
    {
       dtable[i]= 'A'+i;
       dtable[i+9]= 'J'+i;
       dtable[26+i]= 'a'+i;
       dtable[26+i+9]= 'j'+i;
    }
    for(i= 0;i<8;i++)
    {
       dtable[i+18]= 'S'+i;
       dtable[26+i+18]= 's'+i;
    }
    for(i= 0;i<10;i++)
    {
       dtable[52+i]= '0'+i;
    }

    dtable[62]= '+';
    dtable[63]= '/';

    j = 0;
    while(!hiteof)
    {
       byte igroup[3],ogroup[4];
       int c,n;

       igroup[0]= igroup[1]= igroup[2]= 0;
       for(n= 0;n<3;n++){
          if( j < inBuf.size() )
          {
             c = inBuf[j++];
```

```
            } else
            {
               hiteof = true;
               break;
            }
            igroup[n]= (byte)c;
         }
         if(n> 0)
         {
            ogroup[0]= dtable[igroup[0]>>2];
            ogroup[1]= dtable[((igroup[0]&3)<<4)|(igroup[1]>>4)];
            ogroup[2]= dtable[((igroup[1]&0xF)<<2)|(igroup[2]>>6)];
            ogroup[3]= dtable[igroup[2]&0x3F];

            if(n<3)
            {
               ogroup[3]= '=';
               if(n<2)
               {
                  ogroup[2]= '=';
               }
            }
            for(i= 0;i<4;i++)
            {
               outBuf += ogroup[i];
            }
         }
      }
}

string getAuthorization()
{
  string m_encode64,name;
  //You should change name to your own axl server user name and passwd
  //in this example, "CCMAdministrator" is the user name and "cisco_cisco" is the passwd.
  name="CCMAdministrator:cisco_cisco";
  encodeBase64(name,m_encode64);
  return m_encode64;
}

void
BuildDeviceNameSQL(string &buf,  // Buffer to build AXL
                                  string& deviceNumber, // DN
                                  string& seqNum )
{
    const int BUFSIZE = 2048;
    char buff[BUFSIZE];           // Temp buffer
    string strHTTPHeader;         // HTTP/AXL Header
    string strAXLRequest;         // AXL Request

    strAXLRequest = "<SOAP-ENV:Envelope xmlns:SOAP-ENV=";
    strAXLRequest += "\"http://schemas.xmlsoap.org/soap/envelope/\"";
    strAXLRequest += " xmlns:SOAP-ENC=\"http://schemas.xmlsoap.org/soap/encoding/\"";
    strAXLRequest += " xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"";
    strAXLRequest += " xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"> ";
    strAXLRequest += "<SOAP-ENV:Body> ";
    strAXLRequest += "<m:executeSQLQuery xmlns:m=\"http://www.cisco.com/AXL/API/1.0\"
                          sequence=\"" + seqNum + "\"> ";
    strAXLRequest += "<m:sql> ";
    strAXLRequest += "SELECT * FROM Device ";
    strAXLRequest += "</m:sql> ";
    strAXLRequest += "</m:executeSQLQuery> ";
    strAXLRequest += "</SOAP-ENV:Body> ";
    strAXLRequest += "</SOAP-ENV:Envelope>";

    strHTTPHeader = "POST /axl/ HTTP/1.1\r\n";

    strHTTPHeader += "Host: localhost:8443\r\n";
```

```
    strHTTPHeader += "Accept: text/*\r\n";
    strHTTPHeader += "Authorization: Basic ";
    strHTTPHeader += getAuthorization() + "\r\n";
    strHTTPHeader += "Content-type: text/xml\r\n";
    strHTTPHeader += "SOAPAction: \"CUCM:DB ver=6.0\"\r\n";
    strHTTPHeader += "Content-length: ";

    // temporarily use the buffer to store the length of the request
    sprintf( buff, "%d", strAXLRequest.length() );

    strHTTPHeader += buff;
    strHTTPHeader += "\r\nConnection: Keep-Alive";
    strHTTPHeader += "\r\n\r\n";

    // put the HTTP header and SOAP XML together
    buf = strHTTPHeader + strAXLRequest;

    return;

}

int main(int argc, char** argv)
{
  struct sockaddr_in saddr;
  SSL_METHOD *meth;
  SSL_CTX *sslctx;
  SSL *ssl;
  X509* server_cert;
  string buff,line,seqnum;
  char buffer[2048];
  int status,error;
  char *str;

  if( argc!=3 )
  {
     printf("Usage : ssltest <ip> <port> \n");
     printf("Usage : the default port is 8443 \n");
     printf("Usage : the ip is the ip of ccm5.0 \n");
     printf("Example: ssltest 10.77.31.168 8443 \n");

     exit(2);
  }

  int sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
  if(sock<0)
  {
    printf("create socket failed\n");
    exit(1);
  }
  saddr.sin_family=AF_INET;
  saddr.sin_port=htons(atoi(argv[2]));
  saddr.sin_addr.s_addr =inet_addr(argv[1]);
  status=connect(sock,(struct sockaddr *)&saddr,sizeof(saddr));
  if(status<0)
  {
    printf("connect to %s failed\n",argv[1]);
    exit(2);
  }
  SSL_library_init();
  meth=TLSv1_client_method();
  sslctx=SSL_CTX_new(meth);
  if(!sslctx)
  {
     printf("SSL_CTX_new failed\n");
     close(sock);
     exit(3);
  }
```

```
SSL_CTX_set_verify(sslctx,SSL_VERIFY_NONE,NULL);
ssl =SSL_new(sslctx);
if(!ssl)
{
  printf("SSL_new failed\n");
  close(sock);
  exit(4);
}
status=SSL_set_fd(ssl,sock);
if(!status)
{
  printf("SSL_set_fd failed\n");
  close(sock);
  exit(5);
}
SSL_set_mode(ssl,SSL_MODE_AUTO_RETRY);
status=SSL_connect(ssl);
error=SSL_get_error(ssl,status);
switch(error)
{
  case SSL_ERROR_NONE:
    printf("connect successful\n");
    break;
  case SSL_ERROR_ZERO_RETURN:
    printf("peer close ssl connection \n");
    break;
  default:
    printf("connect error is %d\n",error);
 }

 server_cert = SSL_get_peer_certificate (ssl);
 if(!server_cert)
 {
   printf("get server certificate failed!\n");
   SSL_shutdown(ssl);
   close(sock);
   exit(6);
 }
 str= X509_NAME_oneline(X509_get_subject_name (server_cert),0,0);
 if(str)
 {
   printf("subject :%s\n",str);
 }
 else
   printf("subject is empty\n");
 str = X509_NAME_oneline (X509_get_issuer_name  (server_cert),0,0);
 if(!str)
   printf("issuer name is :%s\n",str);
 else
   printf("issuer name is empty \n");
 line="12";
 seqnum="1234";
 BuildDeviceNameSQL(buff,line,seqnum);
 SSL_write(ssl,buff.c_str(),buff.length());
 printf("\n");
 printf("\n");
 printf("Request sent is:\n");
 printf(buff.c_str());
 printf("\n");
 printf("\n");
 SSL_read(ssl,buffer,sizeof(buffer));

 printf("Response from server is: \n%s\n",buffer);
 status=SSL_shutdown(ssl);
 if(status==1)
   printf("shutdown successful\n");
 else
```

```
        printf("\nshutdown error code is %d\n",status);
    close(sock);
}
```

Java Example

This code example uses a hard-coded AXL request and sends it to the AXL server that is running on the local system (localhost). It then reads the response and outputs the response to the screen.

```
import java.io.*;
import java.net.*;
import javax.net.ssl.*;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;

public class AXLJavaClient {
    public static void main(String[] args) {
        byte[] bArray = null; // buffer for reading response from
        Socket socket = null; // socket to AXL server
        OutputStream out = null; // output stream to server
        InputStream in = null; // input stream from server

        String sAXLSOAPRequest = "";
        // HTTPS header and SOAP payload
        String sAXLRequest = null; // will hold only the SOAP payload
        //username=CCMAdministrator and password=cisco_cisco
        String authorization = "CCMAdministrator" + ":" + "cisco_cisco";
        // base64 encoding of the username and password
        authorization = new sun.misc.BASE64Encoder().encode(authorization.getBytes());
        // Form the http header
        sAXLSOAPRequest = "POST /axl/ HTTP/1.0\r\n";
        sAXLSOAPRequest += "Host:localhost:8443\r\n";
        sAXLSOAPRequest += "Authorization: Basic " + authorization + "\r\n";
        sAXLSOAPRequest += "Accept: text/*\r\n";
        sAXLSOAPRequest += "Content-type: text/xml\r\n";
        sAXLSOAPRequest += "SOAPAction: \"CUCM:DB ver=6.0\"\r\n";
        sAXLSOAPRequest += "Content-length: ";

        // Build the SOAP payload
        sAXLRequest = "<SOAP-ENV:Envelope xmlns:SOAP-ENV=\"http://schemas.xmlsoap.org/soap/envelope/\" ";
        sAXLRequest += "xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"> ";
        sAXLRequest += "<SOAP-ENV:Body> <axl:getPhone xmlns:axl=\"http://www.cisco.com/AXL/1.0\" ";
        sAXLRequest += " xsi:schemaLocation=\"http://www.cisco.com/AXL/1.0 http://
ccmserver/schema/axlsoap.xsd\" ";
        sAXLRequest += "sequence=\"1234\"> <phoneName>SEP000000000009</phoneName>";
        sAXLRequest += "</axl:getPhone> </SOAP-ENV:Body> </SOAP-ENV:Envelope>";


        // finish the HTTPS Header
        sAXLSOAPRequest += sAXLRequest.length();
        sAXLSOAPRequest += "\r\n\r\n";

        // now add the SOAP payload to the HTTPS header, which completes the AXL
        // SOAP request
        sAXLSOAPRequest += sAXLRequest;
        try {
            AXLJavaClient axl = new AXLJavaClient();
            // Implement the certificate-related stuffs required for sending request via https
            X509TrustManager xtm = axl.new MyTrustManager();
            TrustManager[] mytm = { xtm };
            SSLContext ctx = SSLContext.getInstance("SSL");
```

```
        ctx.init(null, mytm, null);
        SSLSocketFactory sslFact = (SSLSocketFactory) ctx.getSocketFactory();
        socket = (SSLSocket) sslFact.createSocket("10.77.31.203", Integer.parseInt("8443"));
        in = socket.getInputStream();
        // send the request to the server
        // read the response from the server
        StringBuffer sb = new StringBuffer(2048);
        bArray = new byte[2048];
        int ch = 0;
        int sum = 0;
        out = socket.getOutputStream();
        out.write(sAXLSOAPRequest.getBytes());
        while ((ch = in.read(bArray)) != -1) {
            sum += ch;
            sb.append(new String(bArray, 0, ch));
        }
        socket.close();
        // output the response to the standard output
        System.out.println(sb.toString());
    } catch (UnknownHostException e) {
        System.err.println("Error connecting to host: " + e.getMessage());
        return;
    } catch (IOException ioe) {
        System.err.println("Error sending/receiving from server: " + ioe.getMessage());
        // close the socket
    } catch (Exception ea) {
        System.err.println("Unknown exception " + ea.getMessage());
        return;
    }
    finally{
        try {
            if (socket != null)
                socket.close();
        } catch (Exception exc) {
                exc.printStackTrace();
                System.err.println("Error closing connection to server: "+ exc.getMessage());
        }
    }
}

public class MyTrustManager implements X509TrustManager {

    MyTrustManager() {
        // create/load keystore

    }

    public void checkClientTrusted(X509Certificate chain[], String authType)
            throws CertificateException {

    }

    public void checkServerTrusted(X509Certificate chain[], String authType)
            throws CertificateException {

    }

    public X509Certificate[] getAcceptedIssuers() {

        return null;
    }
}
```

```
}
```

In addition to these examples, refer to the AXL SQL Toolkit, which is available for download from the Cisco Unified Communications Manager server at https://ccmserver:8443/plugins/axlsqltoolkit.zip.

Using executeSQLUpdateAXL

This example illustrates the use of the executeSQLUpdateAXL request:

```
POST :8443/axl/
Host: axl.myhost.com:8443
Accept: text/*
Authorization: Basic bGFycnk6Y3VybHkgYW5kIG1vZQ==
Content-type: text/xml
SOAPAction: "CUCM:DB ver=6.0"
Content-length: 613

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <axlapi:executeSQLUpdate sequence="1"
xmlns:axlapi="http://www.cisco.com/AXL/API/1.0" xmlns:axl="http://www.cisco.com/AXL/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.cisco.com/AXL/API/1.0 axlsoap.xsd">
            <sql>
                insert into device (fkPhoneTemplate,fkDevicePool,tkclass, tkpreemption,
tkdeviceprofile,  tkmodel, tkdeviceprotocol, tkproduct, description,
tkstatus_mlppindicationstatus, name, pkid) values ('Standard 7941 SCCP','default',1, 2, 2,
115, 0, 115, '', 0, 'Cisco 7941', newid())
            </sql>
        </axlapi:executeSQLUpdate>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Using executeSQLQuery

This example illustrates the use of the executeSQLQuery request:

```
POST :8443/axl/
Host: axl.myhost.com:8443
Accept: text/*
Authorization: Basic bGFycnk6Y3VybHkgYW5kIG1vZQ==
Content-type: text/xml
SOAPAction: "CUCM:DB ver=6.0"
Content-length: 613

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <axlapi:executeSQLQuery sequence="1"
xmlns:axlapi="http://www.cisco.com/AXL/API/1.0"
xmlns:axl="http://www.cisco.com/AXL/API/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.cisco.com/AXL/API/1.0 axlsoap.xsd">
            <sql>SELECT * from numplan</sql>
        </axlapi:executeSQLQuery>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Throttling of Requests

The side effects of updating the Cisco Unified Communications Manager database can adversely affect system performance; therefore, the system administrator can control how many AXL requests are allowed to update the database per minute. You can control this value by using the Database Layer Monitor advanced service parameter "MaxAXLWritesPerMinute."

AXL accommodates all requests until the "MaxAXLWritesPerMinute" value is reached. The system rejects subsequent attempts to modify the database with AXL by sending an HTTPS 503 Service Unavailable response. Every minute, AXL resets its internal counter and begins to accept AXL update requests until the limit is reached again.

The following AXL requests, which are considered "Reads," do not count against the preset "MaxAXLWritesPerMinute" service parameter value. All other AXL requests that are not in the following list count against the service parameter value:

- executeSQLQuery
- doDeviceReset
- all AXL "get" requests
- all AXL "list" requests

The execSQLUpdate request is throttled.

AXL Schema Documentation

The axlsqltoolkit.zip plug-in contains the following five AXL schema files:

- AXLAPI.wsdl
- AXLEnums.xsd
- axlmessage.xsd
- axlsoap.xsd
- axl.xsd

These files encapsulate the complete AXL schema, including details of all requests, responses, XML objects, and data types.

In addition to these schema files, two folders exist:

- WSDL-AXIS
- WSDL-NET

Each of these folders contains AXLAPI.wsdl and AXLSoap.xsd files to be used for application development in AXIS or .NET client environments, respectively. The plug-in also contains version specific AXL in folders 1.0 and 6.0.

You can obtain complete documentation of all available AXL messages from the Cisco Developer Services web site: http://www.cisco.com/pcgi-bin/dev_support/access_level/product_support. This website requires a Cisco.com login.

From the Cisco Unified Communications Manager server administration interface, you can use the *Application > Plugins > Cisco Unified Communications Manager AXL SQL Toolkit* command to obtain

- AXL schema (.xsd) files

  See also AXL Schema Documentation, page 1-20.

- The WSDL file

Authentication

The system controls user authentication via the HTTPS Basic Authentication scheme; therefore, you must include the Authorization header in the HTTPS header. Because Base64 encoding takes three 8-bit bytes and represents them as four printable ASCII characters, if the encoded header does not contain an even multiple of four ASCII characters (16, 20, 24, and so on), you must add padding characters (=) to complete the final group of four.

Ensure users are authorized to access AXL. For help with configuring authorization, see Post-Installation Steps and Troubleshooting on the Linux Platform, page 1-22.

If user authentication of the user fails, the system returns an HTTP 401 Access Denied error to the client.

For example, if the user agent wants to send the userid “larry” and password “curly and moe,” it would use the following header field:

```
Authorization: Basic bGFycnk6Y3VybHkgYW5kIG1vZQ==
```

where the string “`bGFycnk6Y3VybHkgYW5kIG1vZQ==`” provides the Base64 encoding of “larry:curly and moe.”

Note The two “equals” characters (=) at the end of the string act as padding characters for Base64 encoding.

Data Encryption

Encrypt AXL SOAP messages by using HTTP Secure Sockets Layer (SSL). SSL remains functional on the web server by default. Ensure AXL requests are made by using the “https” protocol.

Integration Considerations and Interoperability

The AXL API gives much power to developers to modify the Cisco Unified Communications Manager system database. The developer must use caution when using AXL because each API call impacts the system. Abuse of the API can lead to dropped calls and slower system performance. AXL acts as a provisioning and configuration API, not as a real-time API.

If AXL is determined to be using too much CPU time, consider lowering the Database Layer Monitor advanced service parameter MaxAXLWritesPerMinute (see the “Throttling of Requests” section on page 1-20). If this does not solve the problem, consider employing an additional server to be used only by applications that are using AXL.

Note Because AXL is not a real-time API, the autologout function of extension mobility does not work when the user is logged in/out of EM via the AXL interface.

Impact on Backward Compatibility

For Cisco Unified Communications Manager Release 6.0(1), be aware that the defined AXL APIs are backward compatible; however, the executeSQLQuery request, which lets the user run a database query directly on the Cisco Unified Communications Manager database, is **not** backward compatible. Necessary changes in the Cisco Unified Communications Manager 6.0(1) database schema make this true.

- Release 6.0(1) splits some of the tables in the Cisco Unified Communications Manager database. Feature-related information moved to the corresponding dynamic tables. If the direct SQL query to which the 'executeSQLQuery' API referred uses any of the changed tables, you may need to rewrite the query according to the new database schema.
- The columns enduser.password and enduser.pin from the enduser table and the applicationuser.password column from the applicationUser table moved to the credential table as credential.credentials. A direct SQL query that refers to these columns will not work in Cisco Unified Communications Manager Release 6.0(1).

Note Be aware that Cisco Unified Communications Manager password and pin fields are encrypted. Applications should not write to those fields using <executeSQLUpdate>. Instead, update passwords and pins by using the appropriate <updateXXXUser> request.

- The phone API for extension mobility-related parameters has the new tag CurrentConfig. This tag is valid only for the getPhone response. The tag lets AXL provide the original device configuration as well as the logged-in profile information:
  1. If a user has logged in to a device by using a device profile, the CurrentConfig tag contains the values for the extension mobility-related parameters from that device profile.
  2. If no user has logged in, the CurrentConfig tag contains the values of the extension mobility-related parameters for the actual device.
- Schema changes for the CMCInfo and FACInfo APIs help maintain consistency with other AXL APIs. See the description of the changes in AXL APIs Changed for Cisco Unified Communications Manager 6.0(1), page 1-7.

For further details about the Cisco Unified Communications Manager database schema changes, refer to the Cisco Unified Communications Manager *Data Dictionary for Release 6.0(1)*.

Post-Installation Steps and Troubleshooting on the Linux Platform

The system implements AXL as a Java servlet. The Java implementation provides platform independence. AXL accesses the Cisco Unified Communications Manager database by using DBL2, which is a JDBC wrapper implementation. AXL gets packaged as a WAR file. Linux RPM installs the war file for AXL on Cisco Unified Communications Manager server.

Follow the procedures in Post-Installation Steps , page 1-23, to start the AXL service and set up user permissions. Next, follow the procedures in Post-Installation Troubleshooting Checklist, page 1-23, to check the installation.

Post-Installation Steps

You can start or stop the AXL web service from Cisco Unified Communications Manager Serviceability. The service is disabled by default. You should start the service before using the AXL APIs.

Starting the AXL Service

Step 1 From the Cisco Unified Communications Manager Administration window, choose **Navigation > Cisco Unified Communications Manager Serviceability**.

Step 2 Choose **Tools > Service Activation**.

Step 3 From the **Server** box, choose the server and click **GO**.

Step 4 From **Database and Admin Services**, select **Cisco AXL Web Service** and save the changes.

Upon starting the AXL service, AXL gets deployed as a web application within Apache Tomcat. The WAR file gets deployed to Tomcat under /usr/local/thirdparty/jakarta-tomcat/webapps/axl.

Setting AXL API Access Permissions

Step 1 From the Cisco Unified Communications Manager Administration window, choose **User Management > UserGroup > Add New**.

Step 2 To add AXP API access for the new UserGroup

a. Choose **User Management > User Group**.

b. Choose **Role > Assign Role to Group**.

c. Select **Standard AXL API Access**.

d. Click **Add Selected**.

e. On the main page, click **Save**.

Step 3 To add a user to the new UserGroup

a. Choose **User Management > User Group**.

b. Choose **UserGroup > Add End Users to Group**.

c. Select the user and click **Add Selected**.

Post-Installation Troubleshooting Checklist

Use the following checklist to avoid some common problems by fine-tuning your configuration before proceeding with the troubleshooting process:

Step 1 If the AXL client application cannot connect to the AXL service, check the following

- Is the AXL application configured with the correct IP address for the AXL server?
- Is the AXL application configured with the appropriate AXL user credentials?
- Does the application server have HTTPS connectivity to the AXL server?

  Use this URL for accessing AXL: https://system-name:port/axl/ (port is 8443).

- Is HTTPS (secure) configured for AXL?

Step 2 If the AXL functions or requests are failing, check the following

- AXL logs for AXL or SOAP error responses. See Error Codes, page 1-26.
- For further debugging, you can view the AXL log files with the RTMT application.

Step 3 Check that applications in a cluster configuration are connected to the AXL service only on the Cisco Unified Communications Manager Publisher server if the application needs to modify the database.

Step 4 Verify basic AXL functionality by performing the procedure that follows:

1. Go to the AXL API URL via a web browser.

   For instance, enter https://system-name:8443/axl/ in the address text box.

2. When prompted for user name and password, use the standard administrator login, or use the user name that is associated with a user group that is assigned the AXL role.

3. Look for a plain page that states the AXL listener is working and accepting requests but only communicates via POST**.**

This verifies functionality and user access.

Step 5 Check the Cisco Database Layer service parameter MaxAXLWritesPerMinute:

- Updating the Cisco Unified Communications Manager database can have the side effect of adversely affecting system performance. To prevent this effect, the system administrator can control how many AXL requests are allowed to update the database per minute through the Database Layer service parameter MaxAXLWritesPerMinute.
- AXL accommodates all requests until the MaxAXLWritesPerMinute value is reached. The default value of this parameter is 50, and the maximum configurable value is 999; however, the maximum Cisco supported value for production systems is 60.
- After the MaxAXLWritesPerMinute value is reached, the system rejects attempts to modify the database with AXL with an HTTP 503 Service Unavailable response. Every minute, AXL resets its internal counter and begins to accept AXL update requests until the MaxAXLWritesPerMinute value is reached.

Step 6 If the AXL functions or requests are failing with error as User Authorization error: “Access to the requested resource has been denied,” check whether the user has the permission to the Standard AXL API Access. You can check this from the Permission Information section of the EndUser configuration window.

AXL Trace Logs

AXL trace logs contain the text of every AXL request and response, along with user and origination IP information. Trace logs prove useful for identifying who is making AXL requests, inspecting the AXL XML request for format or syntax errors, and determining the actual AXL service response or errors.

The system writes the AXL trace logs to the /var/log/active/tomcat/logs/axl/log4j directory. You can view them with RTMT. File names are axl####.log, where # represents a number from 0000 (zero) to the maximum number of files allowed. The maximum file size is 1 MB by default. The maximum number of stored files defaults to 10. You can change these settings through the Serviceability windows .

Note If an AXL login request contains a <password> tag with an xmlns attribute value, versions of the AXL API prior to 6.0(1) or 5.1(2) log the password in clear text. In later versions of the API, the system replaces the password with an "*" character.

For .NET WSDL applications, including the xmlns attribute in the <password> tag would be typical behavior, and, in earlier releases, this could represent a security issue. If the SOAP message body contains a namespace tag, you do not need to specify the xmlns attribute for each individual tag.

While analyzing the log files,

- Determine which log file is currently active by timestamp.
- Look for Exception traces that indicate processing errors.
- If no traces are being added, verify that Tomcat is running, AXL is currently activated, and a client application is attempting to communicate with the AXL API.

The following sample shows the AXL trace log output:

```
2007-03-17 05:32:26,512 INFO  [http-8443-Processor21] axl.AxlListener - Received request
1173323669700 from CCMAdministrator at IP 10.77.31.203
2007-03-17 05:32:26,513 INFO  [http-8443-Processor21] axl.AxlListener - <!-- edited with
XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by Jerry Vander Voord (Cisco Systems)
--><SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Body>
<axlapi:addGatekeeper sequence="1" xmlns:axlapi="http://www.cisco.com/AXL/API/1.0"
xmlns:axl="http://www.cisco.com/AXL/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.cisco.com/AXL/API/1.0 axlsoap.xsd">
<gatekeeper>
    <name>AXL-Sample-GK1</name>
    <description>This is a sample gatekeeper</description>
    <rrqTimeToLive>30</rrqTimeToLive>
    <retryTimeout>30</retryTimeout>
    <enableDevice>false</enableDevice>
</gatekeeper>
</axlapi:addGatekeeper></SOAP-ENV:Body></SOAP-ENV:Envelope>
2007-03-17 05:32:26,668 INFO  [http-8443-Processor21] axl.Handler - Handler initializing
2007-03-17 05:32:26,788 INFO  [http-8443-Processor21] axl.AxlListener - <?xml
version="1.0" encoding="UTF-8"?><SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:Header/><SOAP
-ENV:Body>
<axl:addGatekeeperResponse xmlns:axl=http://www.cisco.com/AXL/1.0
xmlns:xsi="http://www.cisco.com/AXL/1.0" sequence="1">
<return>{7EB31958-C24A-3E63-3E4B-A8446365D35}</return>
</axl:addGatekeeperResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>
2007-03-17 05:32:26,789 INFO  [http-8443-Processor21] axl.AxlListener - Request
1173323669700 was process in 356ms
```

The AXL trace log contains

- Time that the request was received - 05:32:26,512
- Client IP address - IP 10.77.31.203
- Client user ID - CCMAdministrator
- Request ID number - 1173323669700
- Request contents – addGatekeeper, <gatekeeper>, <name>, and so on
- Response contents - <name>
- Time taken for the response - 356 ms

Follow these steps to set the AXL trace level from the Serviceability window:

Step 1 From the Cisco Unified Communications Manager Administration window, choose **Application > Cisco Unified Communications Manager Serviceability**.

Step 2 Choose **Trace > Configuration**.

Step 3 From the **Servers** column, select the server and click **GO**.

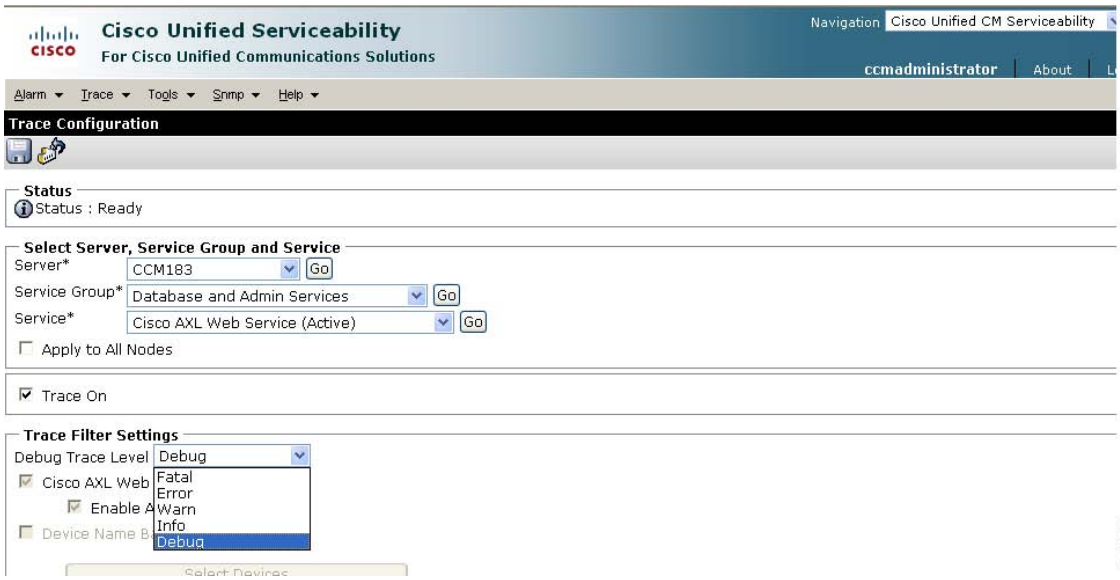
Step 4 From the **Service Group** box, select **Database And Admin Services** and click **GO.**

Step 5 From the **Services** box, select the **Cisco AXL Web Service** and click **GO**.

Step 6 Check the **Trace On** check box.

Step 7 If you want the trace to apply to all Cisco Unified Communications Manager servers in the cluster, select the **Apply to All Nodes** check box.

Step 8 From the **Debug Trace Level** field, select **Debug**.



Step 9 You can set trace output options from the same window.

Note You should enable AXL logs only on request from Cisco TAC or Cisco Developer Services.

Error Codes

If an exception occurs on the server, or if any other error occurs during the processing of an AXL request, the system returns an error in the form of a SOAP Fault message, such as in the following example:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Client</faultcode>
```

```
            <faultstring>
            <![CDATA[
            An error occurred during parsing
            Message: End element was missing the character '>'.

            Source = Line : 41, Char : 6
            Code : c00ce55f, Source Text : </re
            ]]>
            </faultstring>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Fault messages can also contain more detailed information. The following example shows a detailed SOAP Fault message:

```
<SOAP-ENV:Envelope  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" S
OAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Client</faultcode>
          <faultstring>Device not found with name SEP003094C39708.</faultstring>

            <detail  xmlns:axl="http://www.cisco.com/AXL/1.0"
                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:schemaLocation="http://www.cisco.com/AXL/1.0
                                         http://myhost/CCMApi/AXL/V1/axlsoap.xsd">
                <axl:error  sequence="1234">
                    <code>0</code>
                    <message>
<![CDATA[
Device not found with name SEP003094C39708.
]]>
                    </message>
                    <request>doDeviceLogin</request>
                </axl:error>
            </detail>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The <detail> element of a SOAP Fault contains the error codes. The <axl:Error> element represents the errors. If a response to a request contains an <error> element, the user agent can determine the cause of the error by looking at the subelements of the <error> tag.

The following list describes the <error> element.

- The <code> element is a numerical value that the user agent uses to find out what type of error occurred. The next table describes the error codes:

Error Code	Description
5000	Unknown Error: an unknown error occurred while the request was processed. A problem on the server or mistakes in the request can cause this error.
5002	Unknown Request Error: the user agent submitted a request that is unknown to the API.
5003	Invalid Value Exception: the system detected an invalid value in the XML request.

5004	AXL Unavailable Exception: the AXL service was too busy to handle the request at that time. The request should be sent again at a later time.
5005	Invalid Value Exception: the system detected an invalid value in the XML request.

- The <message> element gives the user agent a detailed error message explaining the error.
- The <request> element lets the user agent determine what type of request generated this error. Because this element is optional, it may not always appear.

Using the AXL API with AXIS

This section explains how to use the AXLAPI.wsdl file in a Java programming environment.

Cisco has verified the AXLAPI.wsdl file in the WSDL-AXIS folder in the AXIS environment.

Cisco provides the associated schema file, AXLSoap.xsd, only for code generation. Cisco has modified this file to minimize the backwards compatibility impact of future changes in the database schema. Use the WSDL and the schema files in the parent directory for reference and for validation of responses.

When you run the wsdl2Java utility to create the Java source code by using AXLAPI.wsdl, the utility throws two errors that are specific to AXIS_1_4. For further details on these errors, refer to http://issues.apache.org/jira/browse/AXIS-2545 and http://issues.apache.org/jira/browse/AXIS-1280.

The incorrect ordering of the parameters that are passed to the constructor causes the first AXIS jira error. A code example follows:

```
public class XNPDirectoryNumberShareLineAppearanceCSS  extends
com.cisco.www.AXL.API._6_0.XCallingSearchSpace  implements java.io.Serializable {
>
>
super(
            uuid,
            name,
            description,
            clause,
            dialPlanWizardGenId,
            members);
```

However, the parent constructor is defined as

```
public XCallingSearchSpace(
           org.apache.axis.types.Name name,
           java.lang.String description,
           java.lang.String clause,
           org.apache.axis.types.NonNegativeInteger dialPlanWizardGenId,
           com.cisco.www.AXL.API._6_0.XCallingSearchSpaceMember[] members,
           java.lang.String uuid) {
           this.name = name;
           this.description = description;
           this.clause = clause;
           this.dialPlanWizardGenId = dialPlanWizardGenId;
           this.members = members;
           this.uuid = uuid;
    }
```

You need to change either the constructor or the constructor calling as shown below:

```
super(
            name,
            description,
            clause,
            dialPlanWizardGenId,
            members,
        uuid);
```

The second AXIS jira error relates to having a string constructor for simple types; for example

```
// Simple Types must have a String constructor
    public XLoadInformation(java.lang.String _value) {
    super(_value);
}
```

For such cases, the corresponding schema file (axl.xsd) in the parent schema folder must be referred and must implement the string class that these classes can inherit.

Using the AXL API in a .NET Environment

To integrate the AXL API with a .NET client, you must modify the Cisco-provided WSDL and XSD files.

Cisco provides the associated schema file, AXLSoap.xsd, only for code generation. Cisco has modified this file to minimize the backwards compatibility impact of future changes in the database schema. Use the WSDL and the schema files in the parent directory for reference and for validation of responses.

The inability of .NET to handle complex schemas necessitates some of the changes that are described below.

Required Changes to the Generated Code

After running WSDL.exe, you must make several changes to the generated code. Run WSDL.exe by using the following command:

```
wsdl.exe AXLAPI.wsdl axlsoap.xsd
```

This command generates the file AXLAPIService.cs. The class AXLAPIService in AXLAPIService.cs requires at least three changes:

1. Create an ICertificatePolicy-derived class, which will later be associated with our service. This class represents a brute-force approach to policy and certificate management. You need to use this method in 5.x and 6.x AXL due to the use of HTTPS.

```
public class BruteForcePolicy : System.Net.ICertificatePolicy
{
    public bool CheckValidationResult(System.Net.ServicePoint sp,
           System.Security.Cryptography.X509Certificates.X509Certificate cert,
           System.Net.WebRequest request, int problem)
    {
    return true;
    }
}
```

2. Modify the service constructor to take username/password credentials, with the Cisco Unified Communications Manager IP address as an argument, and associate the BruteForcePolicy class with the static CertificatePolicy manager.

```
public AXLAPIService(string ccmIp, string user, string password)
{
    System.Net.ServicePointManager.CertificatePolicy = new BruteForcePolicy();

    this.Url = "https://" + ccmIp + ":8443/axl/";
    this.Credentials = new System.Net.NetworkCredential(user, password);
}
```

3. The .NET framework uses the *expects* header differently (http://issues.apache.org/bugzilla/show_bug.cgi?id=31567). Several possible workarounds exist to this problem:

   a. Override the GetWebRequest method to use HTTP 1.0 due to an error between TOMCAT/AXIS and the .NET HTTP 1.1 Web Service request mechanism.

```
protected override System.Net.WebRequest GetWebRequest(Uri uri)
{
    System.Net.HttpWebRequest request = base.GetWebRequest (uri) as
        System.Net.HttpWebRequest;
    request.ProtocolVersion = System.Net.HttpVersion.Version10;

    return request;
}
```

   b. Override the GetWebRequest method to manually embed the authentication string. If you do this, do not use the line

```
this.Credentials = new System.Net.NetworkCredential(user, password);
```

   from the constructor that is provided in point 2 earlier in this section.

```
protected override System.Net.WebRequest GetWebRequest(Uri uri)
{
    System.Net.HttpWebRequest request =(System.Net.HttpWebRequest)base.GetWebRequest(uri);
    if (this.PreAuthenticate)
    {
        System.Net.NetworkCredential nc = this.Credentials.GetCredential(uri,"Basic");
        if (nc != null)
        {
            byte[] credBuf = new System.Text.UTF8Encoding().GetBytes(nc.UserName + ":" + nc.Password);
            request.Headers["Authorization"] = "Basic " + Convert.ToBase64String(credBuf);
        }
    }
    return request;
}
```

   c. If you use wsdl2wse (WSE library) instead of wsdl.exe, you cannot override the HTTP version or supply HTTP headers manually. To use WSE, you must set the keepalive header to false for the generated class, or set the user-agent to restricted. This technique will work as an alternative to steps a and b.

Resolving JIT Errors

When you compile and attempt to instantiate the AXLAPIService class, you should expect one error to appear while the types are inspected and loaded.

Class XUserUserGroup includes a field that was generated incorrectly. You must remove one of the '[]' from the two '[][]' brackets after the XUserUserGroupUserRolesUserRole field:

```
public XUserUserGroupUserRolesUserRole[] userRoles;
```

Backward Compatibility Issues

When you add the definitions for new Cisco Unified IP Phone devices to the Cisco Unified Communication Manager database, the original WSDL that was sent out for that Cisco Unified Communication Manager becomes outdated. For example, the XModel enumeration in Cisco CallManager Release 4.1.3 does not contain the Cisco Unified IP Phone 7961G-GE.

However, if you install the latest device pack that contains that device information into your release 4.1.3 environment, that value will be returned if you use the listAllDevices or getPhone commands for that device name. This causes .NET to throw an exception when it encounters the new model because the definition does not contain the mode.

More generally, almost all enumerations in AXLEnums.xsd could change in some future release, which in turn might create backward incompatibility with your code. To address this issue, Cisco has changed the type of all of the tags that use any of these enumerations to String and added an annotation to that tag that specifies where to look for the correct value (AXLEnums.xsd).

Tag Serialization Issues

If you generate the client stub by using wsdl.exe, you may find that some fields that have default values that are defined in the schema would not work if passed in the AXL request. For example, in the updatePhoneReq class of the generated client stub, a field named "ignorePresentationIndicators" has a default value of "False" defined in the schema.

```
[System.Xml.Serialization.XmlTypeAttribute(Namespace="http://www.cisco.com/AXL/6.0")]
    public class UpdatePhoneReq : APIRequest {
        .
        .        .

[System.Xml.Serialization.XmlElementAttribute(Form=System.Xml.Schema.XmlSchemaForm.Unquali
fied)]
        [System.ComponentModel.DefaultValueAttribute(false)]
        public bool ignorePresentationIndicators = false;
        .
        .
    }
```

When this tag is sent with a value of false, XmlSerializer does not serialize this tag because of a design restriction in Microsoft .NET Framework 1.0. Refer to http://support.microsoft.com/kb/325691.

To work around this problem, comment out all instances of

```
[System.ComponentModel.DefaultValueAttribute(XXX)]
```

in the generated client stub as shown:

```
[System.Xml.Serialization.XmlTypeAttribute(Namespace="http://www.cisco.com/AXL/6.0")]
            public class UpdatePhoneReq : APIRequest {
                .
                .                .
[System.Xml.Serialization.XmlElementAttribute(Form=System.Xml.Schema.XmlSchemaForm.Unquali
fied)]
        // Comment this line below
        //[System.ComponentModel.DefaultValueAttribute(false)]
        public bool ignorePresentationIndicators = false;                .
                .
         }
```

A second issue that is found when you are using the version of wsdl.exe that comes with .NET 1.0 is that some tags, including fkcallingsearchspace_autoregistration, do not get updated to null/none in the database.

This appears to be an issue in which .NET does not serialize tags that are defined as nillable=true in the schema.

For example, to work around this limitation for the tag callingSearchSpace in updatePhoneReq in the generated stub, you can remove the "Form=System.Xml.Schema.XmlSchemaForm.Unqualified" from

```
     [System.Xml.Serialization.XmlElementAttribute("name", typeof(string),
Form=System.Xml.Schema.XmlSchemaForm.Unqualified)]
          [System.Xml.Serialization.XmlElementAttribute("uuid", typeof(string),
Form=System.Xml.Schema.XmlSchemaForm.Unqualified)]
          [System.Xml.Serialization.XmlChoiceIdentifierAttribute("ItemElementName")]
          public string Item;
```

With this change, the serializer will serialize the tags. Passing the tag value as "" will set the callingSearchSpace to null/None. The same workaround applies to other such tags.

Names Containing Special Characters

Using the version of wsdl.exe that comes with .NET 1.0, Cisco has found that when attempting to add elements like gatewayEndpoint,MGCPEndpoint or CSS where the name contains special characters, the elements do not get updated in the database properly.

For example, a gatewayEndpoint with name="AALN@SAA000011114444" sent as name="AALN_x0040_SAA000011114444" in the AXL request.

This appears to be a limitation in .NET serialization of tags that are defined as type xsd:Name in the schema.

In the XML specification, the type xsd:name is defined as a token that begins with a letter or one of a few punctuation characters and continues with letters, digits, hyphens, underscores, colons, or periods, together known as name characters. Thus, xsd:name does not allow any special characters such as '@' or '/'.

One workaround involves changing the data type from "Name" to "string" in the generated stub:

Original Code

```
public class XDevice {
    [System.Xml.Serialization.XmlElementAttribute
    (Form=System.Xml.Schema.XmlSchemaForm.Unqualified,DataType="Name")]
    public string name;
```

Modified Code

```
public class XDevice {
    [System.Xml.Serialization.XmlElementAttribute(typeof(string),
        Form=System.Xml.Schema.XmlSchemaForm.Unqualified)]
    public string name;
```

With this modification, the special characters in the name will be updated in the database without any conversion.

Returned Namespace for AXIS and .NET Applications

By default, the AXLAPI.wsdl carries the namespace http://www.cisco.com/AXL/API/6.0. The generated client stubs also have this namespace. In some situations, you must change the namespace in AXLAPI.wsdl before creating the client stubs.

The namespace that is returned in the AXL response depends on two factors:

1. Whether the SOAPAction attribute in the HTTP header had the value "CUCM:DB ver=6.0."
2. The value of the "Send Valid Namespace in AXL Response" service parameter in the Cisco Unified Communications Manager Administration Service Parameter window.

If the SOAPAction attribute in the HTTP header has the value "CUCM:DB ver=6.0":

- The AXL response will have the namespace value that the "Send Valid Namespace in AXL Response" service parameter specifies: either http://www.cisco.com/AXL/API/6.0 or http://www.cisco.com/AXL/6.0.
- If you set the service parameter "Send Valid Namespace in AXL Response" to true, the namespace that is returned in the AXL response will be http://www.cisco.com/AXL/API/6.0, which will match the namespace that is specified in AXLAPI.wsdl.
- If you set this service parameter to False, the namespace that is returned in the AXL response will be http://www.cisco.com/AXL/6.0.

If the SOAPAction attribute has any other value, the AXL response will have the namespace http://www.cisco.com/AXL/API/1.0 or http://www.cisco.com/AXL/1.0, depending on the value of the service parameter.

CHAPTER 2

AXL Serviceability API Programming

This chapter describes the implementation of AXL-Serviceability APIs . Cisco Unified Communications Manager Real-Time information, Performance Counters, and Database information exposure occurs through an AXL-Serviceability interface that conforms to the World Wide Web Consortium (W3C) standard. The Web Service Description Language (WSDL) files provide interface definitions.

To access all AXL API downloads and AXL requests and responses that are found in this document, refer to http://www.cisco.com/pcgi-bin/dev_support/access_level/product_support. You must have a Cisco.com account and password to access this URL.

This chapter covers the following topics:

Introduction

By exposing Cisco Unified Communications Manager real-time information, performance counter, and database information, customers can write customized applications. AXL-Serviceability APIs, extensible SOAP-based XML Web Services, conform to the Simple Object Access Protocol (SOAP) Specification 1.1 and the Web Services Description Language (WSDL) Specification 1.1. AXL-Serviceability APIs represent one server component of the Cisco Unified Communications Manager Serviceability product.

SOAP provides an XML-based communication protocol and encoding format for interapplication communication. SOAP can serve as the backbone to a new generation of cross-platform, cross-language distributed-computing applications, termed Web Services.

AXL-Serviceability APIs provide remote procedure call (RPC) style operations for clients. Clients of AXL-Serviceability APIs can run in different OS platforms and can communicate through the standard SOAP protocol. AXL-Serviceability APIs provide access to core Cisco Unified Communications Manager Serviceability functionality through an open and standard transport protocol and data model.

The AXL-Serviceability interface uses the AXIS 1.4 SOAP server with the AXIS-2250 patch.

Data Model

AXL-Serviceability APIs are based on SOAP with XML request and response messages. APIs must conform to the structure of a SOAP Envelope element. Although SOAP is a standard protocol, many of its data model aspects remain open for flexibility reasons. For example, it permits different transport protocols, such as SMTP, FTP, or HTTP, to carry SOAP messages. The implementation of a SOAP service must specify the bindings of the data model to constitute a concrete wire protocol specification.

The Web Services Description Language (WSDL) Specification 1.1 provides the mechanism to describe the complete SOAP bindings that AXL-Serviceability APIs use.

SOAP Binding

The binding section of the Serviceability SOAP WSDL files specifies AXL-Serviceability API SOAP binding information Binding specifications apply to both request and response messages. SOAP Binding covers the aspects that are explained in the following sections.

Character Encoding

AXL-Serviceability APIs use UTF-8 to encode the data stream in both request and response SOAP messages. The encoding attribute of the XML declaration specifies UTF-8 encoding.
AXL-Serviceability APIs also set “text/xml; charset='utf-8'” as the value of the Content-Type response header field. Internally, AXL-Serviceability APIs processes the data by using UCS-2 Unicode code page.

Binding Style

AXL-Serviceability APIs uses remote procedure call (RPC) binding style. In SOAP, the word operation refers to method or function. Each AXL-Serviceability API operation call gets modeled as an RPC that is encapsulated in SOAP messages. The HTTP request carries RPC calls while the HTTP response carries the call returns. The call information is modeled as a structure. The member elements of the body entry represent the accessor elements that represent the input parameters. The response data is also modeled as a structure with accessors that correspond to output parameters. Parameters that are both input and output must appear in both the request and response message.

Transport Protocols

SOAP allows different transport protocols to carry SOAP messages. AXL-Serviceability APIs use the standard HTTP as its transport. Clients use the POST verb to send requests to AXL-Serviceability APIs.

Note AXL-Serviceability APIs do not use the M-POST method as defined in the HTTP Extension Framework.

Encoding Rule

AXL-Serviceability APIs follow the recommended data model and serialization/encoding rules as defined in Section 5.1 of SOAP Specification 1.1 for both the request and response messages. SOAP simple types are based on the built-in data types that are defined in XML Schema Part 2.

AXL-Serviceability APIs define their own data types, which are derived from the built-in types. The schemas element of the AXL-Serviceability APIs WSDL file specifies AXL-Serviceability APIs that are derived data types. AXL-Serviceability APIs use both simple and compound data types, such as arrays and structures. All operations in AXL-Serviceability APIs pass parameters by value.

For performance reasons, AXL-Serviceability APIs do not specify the size of the array elements in the response message. It leaves the size as [], which means that no particular size is specified, but the clients can determine the size by enumerating the actual number of elements that are inside the array. Point 8 of Section 5.1 of SOAP Specification 1.1 specifies this.

The target namespace URL for AXL-Serviceability API data types schema follows:

http://schemas.xmlsoap.org/soap/envelope/

AXL-Serviceability APIs qualify the first body entry in the response, which represents the call-return, with this target namespace. Similarly, clients of AXL-Serviceability APIs also need to qualify the first body entry in the request, which represents the call, with this namespace.

Request Message

With RPC-style SOAP binding, the request message contains operation call information that is encoded as a struct. The call struct, which appears as the first body entry in the request message, contains the name of the operation and the input parameters. The name of the top-level element represents the operation name. The struct contains accessor element members that represent input parameters. Operations with no parameters have no members in the struct. The names of the accessor elements match the names of the input parameters. The values of the accessor elements represent the values of the input parameters. The order of the accessor elements must match the order of the input parameters as specified in the signature of the operation.

SOAP Action Header

AXL-Serviceability APIs require SOAP clients to include the SOAP Action HTTP header field in the request message. The SOAP 1.1 specification does not place any restrictions on the format of this header. For AXL-Serviceability APIs, the **soapAction** attribute of the SOAP element, which is defined under the binding section of the Serviceability SOAP API WSDL files, specifies the format of the SOAP Action HTTP header. All AXL-Serviceability APIs operations use the following URI format:

"http://schemas.cisco.com/ast/soap/action/#PortName#OperationName"

Note Because the enclosing double-quote characters (" ") are part of the URI, the header must include them.

PortName acts as a placeholder that refers to the name of the port. AXL-Serviceability APIs must support the port. OperationName represents a placeholder that refers to the name of the intended operation within the specified port. This name must match the operation name in the request body, which is specified as the name of the first body entry element. The SOAP Action header indicates the intent of

the SOAP request without having to parse the body of the request. A SOAP server can check the value of this header and determine whether to fail the operation before it proceeds with parsing the XML body. This provides some efficiency on the server side and allows non-SOAP-aware intermediary HTTP servers or proxies to determine the intent of the payload.

Port

A SoapPort basically represents an instantiation of a SoapBinding with a specific network address. Because AXL-Serviceability APIs use HTTP as the transport protocol, the network address in this case specifies an HTTP request URL. SoapPortType, with specific binding rules added to it, provides a basis for the definition of SoapBinding.

The service section of the WSDL file defines the request URL for all AXL-Serviceability API operations. Every request for the AXL-Serviceability APIs operations must use this URL.

SOAP Header

As previously explained, the SOAP header provides a general way to add features to SOAP messages in a decentralized fashion with no prior contract between the sender and recipient. AXL-Serviceability APIs do not use this feature, so no Header element is expected in the envelope and gets ignored. If the Header element gets included with the **mustUnderstand** attribute set to 1, AXL-Serviceability APIs reply with a fault and a fault code value that is set to the **mustUnderstand** value.

The following example shows an AXL-Serviceability API SOAP request message with the HTTP header information:

Example

```
POST  /perfmonservice/services/PerfmonPort HTTP/1.1
Charset: utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
user-agent: ClientName
Host: nozomi
Content-Type: text/xml; charset=utf-8
Content-Length: xxx
SOAPAction: "http://schemas.cisco.com/ast/soap/action/#PerfmonPort#PerfmonOpenSession"

<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <q1:PerfmonOpenSession xmlns:q1="http://schemas.cisco.com/ast/soap/" />
        </soap:Body>
</soap:Envelope>
```

Response Message

For a successful operation call, the call-return gets encoded as a structure. The structure appears as the first body entry of the response. The call-return basically contains the output parameters or return values of the call. The name of the structure top-level element has no significance, and the SOAP 1.1 specification does not place any restriction on the name. The structure contains accessor member elements, which represent the output parameters of the call. The names of the accessor elements match the names of the output parameters. The contents of the accessor elements represent the values of the

output parameters. The order of the accessor elements must match the order of output parameters as specified in the operation signatures. Operation, which does not return any value, contains no member elements in the call-return struct.

AXL-Serviceability APIs use the following naming conventions for the call-return top-level element:

```
SOAPAction: "http://schemas.cisco.com/ast/soap/action/#PortName #OperationName"
```

where `OperationName` represents a placeholder that refers to the name of the operation that is specified in the request message. This format specifically applies only for the AXL-Serviceability APIs implementation.

The target namespace that is described in the "Encoding Rule" section on page 2-3, qualifies the call-return. AXL-Serviceability APIs return HTTP status 200 when the operation succeeds.

For a failed operation call, AXL-Serviceability APIs include the SOAP fault element in the response body. Similar to call-return, the fault element also appears as the first body entry. AXL-Serviceability APIs set HTTP 500 status when sending fault messages.

Fault Message

When an AXL-Serviceability API processes a request and detects that an error occurred, it replies with a SOAP fault element in the response. The fault element appears as the first response body entry. The fault element comprises the following four subelements:

- Fault Code Values
- FaultString
- FaultActor
- Detail

Fault Code Values

AXL-Serviceability APIs use the following standard fault code values as defined in Section 4.4.1 of the SOAP 1.1 specification.

VersionMismatch

This fault code gets set when the namespace URL of the request envelope does not match.

MustUnderstand

This fault code gets set when the clients include the `Header` element in the envelope along with the mustUnderstand attribute set to 1. AXL-Serviceability APIs do not use the Header element. See the "SOAP Header" section on page 2-4 for details.

Client

This fault code gets set when AXL-Serviceability APIs encounters errors that are related to the clients.

Server

This fault code gets set when AXL-Serviceability APIs encounter errors that are related to the service itself. This subelement always exists in the fault element as specified in the SOAP 1.1 specification. This represents a general classification of errors.

FaultString

AXL-Serviceability APIs set a short error description in the faultstring element that is intended for human reading. Similar to faultcode, this subelement is always present as the SOAP 1.1 specification requires. The string value of the FaultString specifically applies only to the AXL-Serviceability APIs.

FaultActor

AXL-Serviceability APIs do not set this subelement. Note that a proxy or intermediary server must include this subelement if it generates a fault message.

Detail

If an AXL-Serviceability API parses and processes the request body and determines that an error occurs afterward, it includes the detailed error information in the detail subelement.

If AXL-Serviceability APIs do not include the detail subelement in the fault element, the error does not relate to the request body. Data types that are defined in the AXL-Serviceability APIs WSDL files specify the encoding rule for the detail subelement.

The following fragments of the file describe the data types:

```
...
64<complexType name='CallInfoType'>
65  <sequence>
66      <element name='FileName' type='xsd:string'/>
67      <element name='LineNo' type='xsd:int'/>
68      <element name='ErrorCode' type='xsd:unsignedInt'/>
69      <element name='Function' type='xsd:string'/>
70      <element name='Params' type='xsd:string'/>
71  </sequence>
72</complexType>
73
74<complexType name='ErrorInfoType'>
75  <sequence>
76      <element name='Version' type='xsd:string'/>
77      <element name='Time' type='xsd:time'/>
78      <element name='ProcId' type='xsd:unsignedInt'/>
79      <element name='ThreadId' type='xsd:unsignedInt'/>
80      <element name='ArrayOfCallInfo' type='tns:ArrayOfCallInfoType'/>
81  /sequence>
82</complexType>
...
128<complexType name='ArrayOfCallInfoType'>
129  <complexContent>
130      <restriction base='SOAP-ENC:Array'>
131          <sequence>
132              <element name='CallInfo'
133                  type='tns:CallInfoType' minOccurs='1' maxOccurs='unbounded'/>
134          </sequence>
135      </restriction>
136  </complexContent>
137</complexType>
```

AXL-Serviceability APIs name the detail entry element as ErrorInfo and the type as ErrorInfoType. This type provides a structure with several accessor elements. The Version accessor contains the build version. The Time accessor denotes the time when the error occurs. The ProcId accessor contains the process ID of the AXL-Serviceability APIs. The ThreadId accessor contains the thread ID that generates the fault. The ArrayOfCallInfo accessor contains an array of CallInfo elements.

The type for CallInfo specifies CallInfoType and also represents a structure. CallInfoType contains detailed information that describes where the error occurs in the code. It also includes the returned error code of the function, and the parameter data. The CallInfoType design allows capturing as much information as needed, so it is easy and fast to track down and investigate a problem. Depending on the implementation of the operation, several CallInfo elements can exist in the array.

The following example shows a successful AXL-Serviceability API SOAP response message with HTTP headers:

Example

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:PerfmonOpenSessionResponse xmlns:m="http://schemas.cisco.com/ast/soap/">
            <SessionHandle>{01944B7E-183F-44C5-977A-F31E3AE59C4C}</SessionHandle>
        </m:PerfmonOpenSessionResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The following example shows a failed AXL-Serviceability API SOAP response message with HTTP headers:

Example

```
HTTP/1.1 500 OK
Content-Type: text/xml; charset=utf-8
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Server</faultcode>
            <faultstring>Perfmon error occurs</faultstring>
            <detail>
                <m:ErrorInfo xmlns:m="http://schemas.cisco.com/ast/soap/">
                    <Version>3.2.0.2</Version>
                    <Time>07/16/2001 - 00:00:24</Time>
                    <ProcId>1200</ProcId>
                    <ThreadId>300</ThreadId>
                    <ArrayOfCallInfo SOAP-ENC:arrayType="m:CallInfoType[]">
                        <CallInfo>
                            <FileName>perfmon.cpp</FileName>
                            <LineNo>396</LineNo>
                            <ErrorCode>3221228473</ErrorCode>
                            <Function>AddCounter</Function>
                            <Params>\\nozomi\tcp\Bad Counter Name</Params>
                        </CallInfo>
                    </ArrayOfCallInfo>
                </m:ErrorInfo>
            </detail>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Namespaces

AXL-Serviceability APIs use the following XML namespaces:

- `http://schemas.xmlsoap.org/soap/envelope/`

  The namespace URI for the SOAP envelope
- `http://schemas.xmlsoap.org/soap/encoding/`

  The namespace for the SOAP-recommended encoding rule that is based on XML Schema
- `http://schemas.cisco.com/ast/soap/`

  The namespace URL for AXL-Serviceability API data types as defined in the WSDL file

Downloading Serviceability SOAP WSDL Files

You can download the Cisco Unified Communications Manager Release 6.0(1) serviceability SOAP WSDL files from the Cisco Unified Communications Manager server directly by simply entering a URL on the web browser. In each of the following examples “servername” must be replaced by an appropriate server IP address.

PerfmonPort SOAP requests service definitions are located at

https://servername:8443/perfmonservice/services/PerfmonPort?wsdl

RisPort SOAP requests service definitions are located at

https://servername:8443/realtimeservice/services/RisPort?wsdl

LogCollectionService SOAP requests service definitions are located at

https://servername:8443/logcollectionservice/services/LogCollectionPort?wsdl

DimeGetFile SOAP requests service definitions are located at

https://servername:8443/logcollectionservice/services/DimeGetFileService?wsdl

ControlCenterServices SOAP requests service definitions are located at

https://servername:8443/controlcenterservice/services/ControlCenterServicesPort?wsdl

SOAPMonitorService SOAP requests service definitions are located at

https://servername:8443/realtimeservice/services/SOAPMonitorService?wsdl

Clients of AXL-Serviceability APIs must download these files to know what services are available, how to form the request message, and how to interpret the response message properly. Basically, the WSDL file has what you need to know about AXL-Serviceability APIs.

Monitoring SOAP Activity

You can use AXIS SOAPMonitor to monitor SOAP activities. Point your browser to

https://servername:8443/realtimeservice/SOAPMonitor

where “servername” is replaced with an appropriate server IP address.

AXL-Serviceability PerfmonPort

PerfmonPort comprises several operations that allow clients to do the following perfmon-related tasks:

- Collect perfmon counter data.

  AXL-Serviceability APIs provide two ways to collect perfmon data: session-based and single-transaction.
- Get a list of all perfmon objects and counter names that are installed in a particular host.
- Get a list of the current instances of a perfmon object.
- Get textual description of a perfmon counter.

PerfmonListCounter

This operation returns the list of Perfmon objects and counters in a particular host.

Request Format

The PerfmonListCounter operation takes a single parameter:

- **Host**
  The type is xsd:string. The Host parameter contains the name or address of the target server from which the client wants to get the counter information.

The following example shows a PerfmonListCounter request:

Example

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <q1:PerfmonListCounter xmlns:q1="http://schemas.cisco.com/ast/soap/">
            <Host xsi:type="xsd:string">nozomi</Host>
        </q1:PerfmonListCounter>
    </soap:Body>
</soap:Envelope>
```

Response Format

PerfmonListCounter returns information that describes the hierarchical structure of Perfmon objects and counters. The body entry includes an ArrayOfObjectInfo element. The following fragments from the AXL-Serviceability APIs WSDL file describe the types that this response uses:

```
...
106 <complexType name='ArrayOfObjectInfoType'>
107     <complexContent>
106         <restriction base='SOAP-ENC:Array'>
109             <sequence>
110                 <element name='ObjectInfo'
111                     type='tns:ObjectInfoType' minOccurs='1' maxOccurs='unbounded'/>
112             </sequence>
113         </restriction>
114     </complexContent>
115 </complexType>
```

The ArrayOfObjectInfo element comprises an array of ObjectInfo elements that have the following type:

```
...
56 <complexType name='ObjectInfoType'>
57     <sequence>
58         <element name='Name' type='tns:ObjectNameType'/>
59         <element name='MultiInstance' type='xsd:boolean'/>
60         <element name='ArrayOfCounter' type='tns:ArrayOfCounterType'/>
61     </sequence>
62 </complexType>

...
24 <simpleType name='ObjectNameType'>
25     <restriction base='string'/>
26 </simpleType>
```

The Name element, whose type is derived from string, describes the name of the object. MultiInstance element indicates whether the object has more than one instance. The ArrayOfCounter element acts as a container for an array of Counter elements that have the following types:

```
...
44 <complexType name='CounterType'>
45     <sequence>
46         <element name='Name' type='tns:CounterNameType'/>
47     </sequence>
48 </complexType>
32 <simpleType name='CounterNameType'>
33     <restriction base='string'/>
34 </simpleType>
```

The Name element, whose type is derived from xsd:string, describes the name of the counter.

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:PerfmonListCounterResponse xmlns:m="http://schemas.cisco.com/ast/soap/">
            <ArrayOfObjectInfo SOAP-ENC:arrayType="m:ObjectInfoType[]">
                <ObjectInfo>
                    <Name>.NET CLR Memory</Name>
                    <MultiInstance>true</MultiInstance>
                    <ArrayOfCounter SOAP-ENC:arrayType="m:CounterType[]">
                        <Counter>
                            <Name># Gen 0 Collections</Name>
                        </Counter>
                        <Counter>
                            <Name># Gen 1 Collections</Name>
                        </Counter>
                        ...
                    </ArrayOfCounter>
                </ObjectInfo>
                <ObjectInfo>
                    <Name>.NET CLR LocksAndThreads</Name>
                    <MultiInstance>true</MultiInstance>
                    <ArrayOfCounter SOAP-ENC:arrayType="m:CounterType[]">
                        <Counter>
                            <Name>Total # of Contentions</Name>
                        </Counter>
                        <Counter>
```

```
                        <Name>Contention Rate / sec</Name>
                    </Counter>
                    <Counter>
                        <Name>Current Queue Length</Name>
                    </Counter>
                    ...
                </ArrayOfCounter>
            </ObjectInfo>
            ...
        </ArrayOfObjectInfo>
    </m:PerfmonListCounterResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

PerfmonListInstance

This operation returns a list of instances of a perfmon object in a particular host. Instances of an object can dynamically come and go, and this operation returns the most recent list.

Request Format

The PerfmonListIntance operation takes the following parameters:

- **Host**
  The type is xsd:string. The Host parameter contains the name or address of the target server on which the object resides.
- **Object**
  The type is xsd:string. The Object parameter contains the name of the object.

The following example shows a PerfmonListInstance request with “nozomi” as the host and “Process” as the object parameter.

Example

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <q1:PerfmonListInstance xmlns:q1="http://schemas.cisco.com/ast/soap/">
            <Host xsi:type="xsd:string">nozomi</Host>
            <Object xsi:type="xsd:string">Process</Object>
        </q1:PerfmonListInstance>
    </soap:Body>
</soap:Envelope>
```

Response Format

PerfmonListInstance returns an element that named ArrayOfInstance. The type for this element specifies ArrayOfInstanceType, which is an array of Instance elements. The following fragments from AXL-Serviceability APIs .WSDL file explain the types that this response uses:

```
...
 84<complexType name='ArrayOfInstanceType'>
 85  <complexContent>
 86      <restriction base='SOAP-ENC:Array'>
 87          <sequence>
```

```
88             <element name='Instance'
89                 type='tns:InstanceType' minOccurs='0' maxOccurs='unbounded'/>
90         </sequence>
91     </restriction>
92 </complexContent>
93</complexType>
```

Note ArrayOfInstanceType can have 0 (zero) Instance elements, in which case the requested object is not of a multi-instance object.

```
...
50<complexType name='InstanceType'>
51 <sequence>
52     <element name='Name' type='tns:InstanceNameType'/>
53 </sequence>
54</complexType>
```

The type for Instance element specifies InstanceType. It represents a structure with a single-element member: Name.

```
...
28<simpleType name='InstanceNameType'>
29 <restriction base='string'/>
30</simpleType>
```

The Name element, whose type is InstanceNameType, which is derived from xsd:string, contains the name of the instance of the requested object.

The following example shows the response to the request in the previous example:

Example

```
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:PerfmonListInstanceResponse xmlns:m="http://schemas.cisco.com/ast/soap/">
            <ArrayOfInstance SOAP-ENC:arrayType="m:InstanceType[]">
                <Instance>
                    <Name>Idle</Name>
                </Instance>
                <Instance>
                    <Name>System</Name>
                </Instance>
                <Instance>
                    <Name>SMSS</Name>
                </Instance>
                <Instance>
                    <Name>CSRSS</Name>
                </Instance>
                <Instance>
                    <Name>WINLOGON</Name>
                </Instance>
                <Instance>
                    <Name>SERVICES</Name>
                </Instance>
                <Instance>
                    <Name>_Total</Name>
                </Instance>
                ...
            </ArrayOfInstance>
        </m:PerfmonListInstanceResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

PerfmonQueryCounterDescription

This operation returns the help text of a particular counter.

Request Format

PerfmonQueryCounterDescription takes the following parameter:

- **Counter**—The name of the counter; type is CounterNameType, which is derived from xsd:string.

The following example shows the PerfmonQueryCounterDescription request:

Example

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <q1:PerfmonQueryCounterDescription
            xmlns:q1="http://schemas.cisco.com/ast/soap/">
            <Counter xsi:type="xsd:string">\\nozomi\Server\Files Open</Counter>
        </q1:PerfmonQueryCounterDescription>
    </soap:Body>
</soap:Envelope>
```

Response Format

PerfmonQueryCounterDescription returns an element that is named `HelpText` that is of the `xsd:string` type. It contains the help text of the requested counter.

The following example shows the response to the request in the previous example.

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:PerfmonQueryCounterDescriptionResponse
            xmlns:m="http://schemas.cisco.com/ast/soap/">
            <HelpText>The number of files currently opened in the server. Indicates
current server
activity.
</HelpText>
        </m:PerfmonQueryCounterDescriptionResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

PerfmonOpenSession

Client programs submit this operation to get a session handle from the AXL-Serviceability APIs. The client needs a session handle to do the session-based perfmon counter data collection. The session handle represents a universally unique identifier that is used once, which guarantees that no duplicate handles exist. AXL-Serviceability APIs keep the opened handles in cache. If no activity occurs on a handle for 25 hours, the AXL-Serviceability API removes the handle and renders it invalid.

In a session-based perfmon data collection, use the following related operations:

- PerfmonOpenSession
- PerfmonAddCounter
- PerfmonRemoveCounter
- PerfmonCollectSessionData
- PerfmonCloseSession

After a client gets a session handle, it normally proceeds to submit the PerfmonAddCounter operation and then follows with the PerfmonCollectSessionData operation. PerfmonCollectSessionData specifies the main operation that collects perfmon data for the clients. When the client no longer needs the session handle, it should submit PerfmonCloseSession, so the AXL-Serviceability APIs can remove the handle from cache. Clients can dynamically add new counters to the session handle and remove counters from it by using the PerfmonRemoveCounter operation while the session handle is still open.

Request Format

The PerfmonOpenSession operation takes no parameter.

The following example shows the PerfmonOpenSession request:

Example

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <q1:PerfmonOpenSession xmlns:q1="http://schemas.cisco.com/ast/soap/" />
        </soap:Body>
</soap:Envelope>
```

Response Format

PerfmonOpenSession returns a single element that is named SessionHandle. Its type specifies SessionHandleType, which is derived from xsd:string, and it contains the guide for the session handle.

The following example shows the PerfmonOpenSession response:

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:PerfmonOpenSessionResponse xmlns:m="http://schemas.cisco.com/ast/soap/">
            <SessionHandle>{01944B7E-183F-44C5-977A-F31E3AE59C4C}</SessionHandle>
        </m:PerfmonOpenSessionResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

PerfmonAddCounter

This operation adds an array of counters to a session handle.

Request Format

PerfmonAddCounter takes the following parameters:

- **SessionHandle**
  The type is SessionHandleType, which is derived from xsd:string. It contains the session handle that the previous PerfmonOpenSession operation previously opened.
- **ArrayOfCounter**
  The type for this element is ArrayOfCounterType, which is an array of Counter elements. Each Counter element contains the name of a counter to be added to the session handle.

The following fragments from AXL-Serviceability APIs describe the types that this request uses:

```
...
95<complexType name='ArrayOfCounterType'>
96  <complexContent>
97      <restriction base='SOAP-ENC:Array'>
98          <sequence>
99              <element name='Counter'
100                 type='tns:CounterType' minOccurs='1' maxOccurs='unbounded'/>
101         </sequence>
102     </restriction>
103 </complexContent>
104</complexType>
```

Note ArrayOfCounterType expects at least one Counter element in the array.

```
...
44<complexType name='CounterType'>
45  <sequence>
46      <element name='Name' type='tns:CounterNameType'/>
47  </sequence>
48</complexType>
```

CounterType represents a structure, and it has a single element member: Name.

```
...
32<simpleType name='CounterNameType'>
33  <restriction base='string'/>
34</simpleType>
```

The Name element that is of string-derived type contains the name of the counter.

The following example shows the PerfmonAddCounter request with two counters. This example uses a single-reference accessor.

Example

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tns="http://tempuri.org/"
    xmlns:types="http://tempuri.org/encodedTypes"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```
        <q1:PerfmonAddCounter xmlns:q1="http://schemas.cisco.com/ast/soap/">
            <SessionHandle xsi:type="xsd:string">
                {1A490F1E-D82C-403F-9CF0-C4D4ABD6FF3E}
            </SessionHandle>
            <ArrayOfCounter soapenc:arrayType="q1:CounterType[2]">
                <Counter>
                    <Name>\\nozomi\process(inetinfo)\handle count</Name>
                </Counter>
                <Counter>
                    <Name>\\nozomi\process(csrss)\handle count</Name>
                </Counter>
            </ArrayOfCounter>
</q1:PerfmonAddCounter">
    </soap:Body>
</soap:Envelope
```

The following shows an example of the PerfmonAddCounter request with three counters in the ArrayOfCounter parameter. This example uses multireference accessors.

Example

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <q1:PerfmonAddCounter xmlns:q1="http://schemas.cisco.com/ast/soap/">
            <SessionHandle xsi:type="xsd:string">
                {1A490F1E-D82C-403F-9CF0-C4D4ABD6FF3E}
            </SessionHandle>
            <ArrayOfCounter href="#id1"/>
        </q1:PerfmonAddCounter>
        <soapenc:Array id="id1"
            xmlns:q2="http://schemas.cisco.com/ast/soap/"
            soapenc:arrayType="q2:CounterType[3]">
            <Item href="#id2" />
            <Item href="#id3" />
            <Item href="#id4" />
        </soapenc:Array>
        <q3:CounterType id="id2" xsi:type="q3:CounterType"
            xmlns:q3="http://schemas.cisco.com/ast/soap/">
            <Name xsi:type="xsd:string">\\nozomi\tcp\Segments/sec</Name>
        </q3:CounterType>
        <q4:CounterType id="id3" xsi:type="q4:CounterType"
            xmlns:q4="http://schemas.cisco.com/ast/soap/">
            <Name xsi:type="xsd:string">\\nozomi\tcp\Connections Reset</Name>
        </q4:CounterType>
        <q5:CounterType id="id4" xsi:type="q5:CounterType"
            xmlns:q5="http://schemas.cisco.com/ast/soap/">
            <Name xsi:type="xsd:string">\\nozomi\tcp\Connections Active</Name>
        </q5:CounterType>
    </soap:Body>
</soap:Envelope>
```

Response Format

The following example shows that the PerfmonAddCounter returns no output:

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:PerfmonAddCounterResponse xmlns:m="http://schemas.cisco.com/ast/soap/"/>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

If PefmonAddCounter fails to add one or more counters that are specified in the request, AXL-Serviceability APIs reply with a fault response. Some counters that are specified in the request may get successfully added, while others failed to be added.

In this case, AXL-Serviceability APIs reply with a fault. The Params element of CallInfo element specifies each failed counter name. Client programs can conclude that counter names, which are specified in the request but do not appear in the fault message, actually get added successfully to the query handle.

PerfmonRemoveCounter

This operation removes an array of counters from a session handle.

Request Format

PerfmonRemoveCounter takes the following parameters:

- **SessionHandle**
  The type is SessionHandleType which is derived from xsd:string. It contains the session handle that the PerfmonOpenSession operation opened previously.
- **ArrayOfCounter**
  The type for this element is ArrayOfCounterType, which is an array of Counter elements. Each Counter element contain the name of a counter to be added to the session handle.

The following example shows a PerfmonRemoveCounter request with three counters in the ArrayOfCounter parameter. This example uses single-reference accessor style.

Example

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tns="http://tempuri.org/"
    xmlns:types="http://tempuri.org/encodedTypes"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <q1:PerfmonRemoveCounter xmlns:q1="http://schemas.cisco.com/ast/soap/">
            <SessionHandle xsi:type="xsd:string">
                {1A490F1E-D82C-403F-9CF0-C4D4ABD6FF3E}
            </SessionHandle>
            <ArrayOfCounter soapenc:arrayType="q1:CounterType[2]">
                <Counter>
```

```
                    <Name>\\nozomi\process(inetinfo)\handle count</Name>
                </Counter>
                <Counter>
                    <Name>\\nozomi\process(csrss)\handle count</Name>
                </Counter>
                <Counter>
                    <Name>\\nozomi\process(regsvc)\handle count</Name>
                </Counter>
            </ArrayOfCounter>
</q1:PerfmonRemoveCounter">
    </soap:Body>
</soap:Envelope
```

The following example shows a PerfmonRemoveCounter that uses multireference accessors.

Example

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <q1:PerfmonRemoveCounter xmlns:q1="http://schemas.cisco.com/ast/soap/">
            <SessionHandle xsi:type="xsd:string">
                {1A490F1E-D82C-403F-9CF0-C4D4ABD6FF3E}
            </SessionHandle>
            <ArrayOfCounter href="#id1" />
        </q1:PerfmonRemoveCounter>
        <soapenc:Array id="id1" xmlns:q2="http://schemas.cisco.com/ast/soap/"
            soapenc:arrayType="q2:CounterType[3]">
            <Item href="#id2" />
            <Item href="#id3" />
            <Item href="#id4" />
        </soapenc:Array>
        <q3:CounterType id="id2" xsi:type="q3:CounterType"
            xmlns:q3="http://schemas.cisco.com/ast/soap/">
            <Name xsi:type="xsd:string">\\nozomi\tcp\Segments/sec</Name>
        </q3:CounterType>
        <q4:CounterType id="id3" xsi:type="q4:CounterType"
            xmlns:q4="http://schemas.cisco.com/ast/soap/">
            <Name xsi:type="xsd:string">\\nozomi\tcp\Connections Reset</Name>
        </q4:CounterType>
        <q5:CounterType id="id4" xsi:type="q5:CounterType"
            xmlns:q5="http://schemas.cisco.com/ast/soap/">
            <Name xsi:type="xsd:string">\\nozomi\tcp\Connections Active</Name>
        </q5:CounterType>
    </soap:Body>
</soap:Envelope>
```

Response Format

PerfmonRemoveCounter returns no data in the response as shown by the following example:

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
    <m:PerfmonRemoveCounterResponse xmlns:m="http://schemas.cisco.com/ast/soap/"/>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

If the PefmonRemoveCounter operation fails to remove one or more counters that the request specifies, the AXL-Serviceability API replies with a fault response with semantics similar to PerfmonAddCounter. If some of the counters that are specified in the request get removed successfully, while others failed to be removed, the AXL-Serviceability API replies with a fault. The Params element of CallInfo element specifies each failed counter name. Client programs can conclude that counter names, which are specified in the request but do not appear in the fault message, actually get removed successfully from the query handle.

PerfmonCollectSessionData

This operation collects the perfmon data for all counters that have been added to the query handle.

Request Format

PerfmonCollectSessionData takes the following parameter:

- **SessionHandle**
  The type is SessionHandleType which is derived from xsd:string. It contains the session handle that the PerfmonOpenSession operation opened previously.

The following example shows a PerfmonCollectSessionData request:

Example

```
<?xml version="1.0" encoding="utf-8" ?>
    <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <q1:PerfmonCollectSessionData xmlns:q1="http://schemas.cisco.com/ast/soap/">
            <SessionHandle xsi:type="xsd:string">
                {FB343D6D-AA6E-4EDB-9CFA-63A5A3ED6405}
            </SessionHandle>
        </q1:PerfmonCollectSessionData>
    </soap:Body>
</soap:Envelope>
```

Response Format

The PerfmonCollectSessionData operation returns the ArrayOfCounterInfo element that contains the value and status of all counters that were previously added to the session handle. The type for ArrayOfCounterInfo element specifies ArrayOfCounterInfoType, which is an array of CounterInfo elements.

The following fragments from AXL-Serviceability APIs .WSDL show the types that this response uses:

```
...
117<complexType name='ArrayOfCounterInfoType'>
118  <complexContent>
119      <restriction base='SOAP-ENC:Array'>
120          <sequence>
121              <element name='CounterInfo'
122                  type='tns:CounterInfoType' minOccurs='1' maxOccurs='unbounded'/>
123          </sequence>
124      </restriction>
125  </complexContent>
126</complexType>
```

ArrayOfCounterInfoType has one or more CounterInfo elements in the array. The CounterInfo element includes the following type:

```
...
36<complexType name='CounterInfoType'>
37  <sequence>
38      <element name='Name' type='tns:CounterNameType'/>
39      <element name='Value' type='xsd:long'/>
40      <element name='CStatus' type='xsd:unsignedInt'/>
41  </sequence>
42</complexType>
...
32<simpleType name='CounterNameType'>
33  <restriction base='string'/>
34</simpleType>
```

CounterInfoType specifies a structure with the following three element members.

- Name

  A CounterNameType, derived from xsd:string, that contains the name of the counter that was previously added to the session handle.

- Value

  A 64-bit signed integer (xsd:long) that contains the value of the counter.

- CStatus

  Indicates whether the value of the counter was successfully retrieved. The type specifies a 32-bit unsigned integer (xsd:unsignedInt). First, check for the value of CStatus element before reading the Value element. If the value of CStatus equals 0 or 1, the Value element contains a good counter value. Otherwise, it indicates a failure in retrieving the counter value; ignore the Value element.

The following example shows a PerfmonCollectSessionData response:

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
<m:PerfmonCollectSessionDataResponse
xmlns:m="http://schemas.cisco.com/ast/soap/">
            <ArrayOfCounterInfo SOAP-ENC:arrayType="m:CounterInfoType[]">
                <CounterInfo>
                    <Name>\\nozomi\tcp\Segments/sec</Name>
                    <Value>0</Value>
                    <CStatus>0</CStatus>
                </CounterInfo>
                <CounterInfo>
                    <Name>\\nozomi\tcp\Connections Reset</Name>
                    <Value>6</Value>
                    <CStatus>0</CStatus>
                </CounterInfo>
                <CounterInfo>
                    <Name>\\nozomi\tcp\Connections Active</Name>
                    <Value>23</Value>
                    <CStatus>0</CStatus>
                </CounterInfo>
            </ArrayOfCounterInfo>
        </m:PerfmonCollectSessionDataResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

PerfmonCloseSession

This operation closes the session handle that the PerfmonOpenSession previously retrieved.

Request Format

PerfmonCloseSession takes the following parameter:

- **SessionHandle**
  The type is SessionHandleType which is derived from xsd:string. It contains the session handle that the PerfmonOpenSession operation previously opened.

The following example shows a PerfmonCloseSession request:

Example

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <q1:PerfmonCloseSession xmlns:q1="http://schemas.cisco.com/ast/soap/">
<SessionHandle xsi:type="xsd:string">
{01944B7E-183F-44C5-977A-F31E3AE59C4C}
</SessionHandle>
        </q1:PerfmonCloseSession>
    </soap:Body>
</soap:Envelope>
```

Response Format

The following example shows that the PerfmonCloseSession does not return data in the response:

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:PerfmonCloseSessionResponse xmlns:m="http://schemas.cisco.com/ast/soap/"/>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

PerfmonCollectCounterData

This operation returns the perfmon data for all counters that belong to an object in a particular host. Unlike the session-based perfmon data collection, this operation collects all data in a single request/response transaction. If the object represents multiple-instance object, this operation always returns the most current instances of the object.

Request Format

PerfmonCollectCounterData takes the following parameters:

- **Host**
  The type is xsd:string. It contains the address of the target server from which the client wants to get the counter information.
- **Object**
  The type is ObjectNameType, which is derived from xsd:string. It contains the name of the perfmon object.

The following example shows a PerfmonCollectCounterData request:

Example

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <q1:PerfmonCollectCounterData xmlns:q1="http://schemas.cisco.com/ast/soap/">
            <Host xsi:type="xsd:string">nozomi</Host>
            <Object xsi:type="xsd:string">Processor</Object>
        </q1:PerfmonCollectCounterData>
    </soap:Body>
</soap:Envelope>
```

Response Format

PerfmonCollectCounterData returns an ArrayOfCounterInfo element, which is an array of CounterInfo elements. CounterInfoType specifies a structure with the following three element members.

- **Name**

  A CounterNameType, derived from xsd:string, that contains the name of the counter that was previously added to the session handle.
- **Value**

  A 64-bit signed integer (xsd:long) that contains the value of the counter.
- **CStatus**

  Indicates whether the value of the counter was successfully retrieved. The type specifies a 32-bit unsigned integer (xsd:unsignedInt). First, check for the value of CStatus element before reading the Value element. If the value of CStatus equals to 0 or 1, the Value element contains a good counter value. Otherwise, it indicates a failure in retrieving the counter value; ignore the Value element.

The following example shows a PerfmonCollectCounterData response:

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:PerfmonCollectCounterDataResponse
            xmlns:m="http://schemas.cisco.com/ast/soap/">
            <ArrayOfCounterInfo SOAP-ENC:arrayType="m:CounterInfoType[]">
                <CounterInfo>
                    <Name>\\nozomi\Processor(0)\% Processor Time</Name>
                    <Value>99</Value>
                    <CStatus>0</CStatus>
                </CounterInfo>
                <CounterInfo>
```

```
                    <Name>\\nozomi\Processor(0)\% User Time</Name>
                    <Value>0</Value>
                    <CStatus>0</CStatus>
                </CounterInfo>
                <CounterInfo>
                    <Name>\\nozomi\Processor(0)\% Privileged Time</Name>
                    <Value>0</Value>
                    <CStatus>0</CStatus>
                </CounterInfo>
                <CounterInfo>
                    <Name>\\nozomi\Processor(0)\Interrupts/sec</Name>
                    <Value>525</Value>
                    <CStatus>0</CStatus>
                </CounterInfo>
                ...
            </ArrayOfCounterInfo>
        </m:PerfmonCollectCounterDataResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Real-Time Information (RisPort)

Selecting Real-Time Information

Request Format

SOAP Action and Envelope Information

This operation allows clients to perform Cisco Unified Communications Manager device-related queries. HTTP header should have following SOAP action for these queries.

```
SOAPAction: "http://schemas.cisco.com/ast/soap/action/#RisPort#SelectCmDevice"
```

Query information includes an Envelope as follows:

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
3.  xmlns:tns="http://schemas.cisco.com/ast/soap/"
4.  xmlns:types="http://schemas.cisco.com/ast/soap/encodedTypes"
5.  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6.  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

Session ID

This SOAP header will have session ID that is a unique session ID from client.

```
7.  <soap:Header>
8.  <tns:AstHeader id="id1">
9.  <SessionId xsi:type="xsd:string">SessionId</SessionId>
10. </tns:AstHeader>
11. </soap:Header>
```

Selection Criteria

Selection criteria type, which is a Cisco Unified Communications Manager Selection criteria, follows the SOAP header.

```
12. <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
13. <tns:SelectCmDevice>
```

If the same information is queried over and over again, send Stateinfo from the previous request for each repetitive query by client.

```
14. <StateInfo xsi:type="xsd:string" />
15. <CmSelectionCriteria href="#id1"/>
16. </tns:SelectCmDevice><tns:CmSelectionCriteria id="id1"
xsi:type="tns:CmSelectionCriteria">
```

Maximum Devices Specification

This example specifies how many maximum devices can be returned for search criteria.

```
17. <MaxReturnedDevices xsi:type="xsd:unsignedInt">10</MaxReturnedDevices>
```

Search Device Classes

This example specifies the device class type to query for real-time status. Device classes include 'Any', 'Phone', 'Gateway', 'H323', 'Cti', 'VoiceMail', 'MediaResources', 'HuntList', 'SIPTrunk', and 'unknown'.

```
18. <Class xsi:type="tns:DeviceClass">Any</Class>
```

This example specifies the Model of the device—255 specifies all models.

```
19. <Model xsi:type="xsd:unsignedInt">255</Model>
```

Device Status in Search Criteria

Specify registered/unregistered status devices. The following example shows status 'Any', 'Registered', 'UnRegistered', 'Rejected', and 'Unknown.'

```
20. <Status xsi:type="tns:CmDevRegStat">Registered</Status>
```

The following example specifies the server name where the search needs to be performed. If no name is specified, a search in all servers in a cluster results.

```
21. <NodeName xsi:type="xsd:string" />
```

Specify Selection type whether it is IP Address/Name

```
22. <SelectBy xsi:type="tns:CmSelectBy">Name</SelectBy>
```

Array of Items for Which Search Criteria Are Specified

The following example specifies an array that contains the IP Address or Device Name of the items for which a real-time status is required.

```
23. <SelectItems href="#id2" />Name or IP</tns:CmSelectionCriteria
24. <soapenc:Array id="id2" soapenc:arrayType="tns:SelectItem[2]">
25. <Item href="#id3"/><Item xsi:null="1"/>
26. </soapenc:Array>
27. <tns:SelectItem id="id3" xsi:type="tns:SelectItem">
28. <Item xsi:type="xsd:string"/></tns:SelectItem>
29. </soap:Body>
30. </soap:Envelope>
```

Response Format

The Response follows the following schema and contains information for one to many servers for each server and contains a sequence of search information that is found on the search criteria.

```
<complexType name='SelectCmDeviceResult'>
      <sequence>
            <element name='TotalDevicesFound' type='xsd:unsignedInt'/>
            <element name='CmNodes' type='tns:CmNodes'/>
      </sequence>
</complexType>
```

CMNodes provides a list of Unified CMNodes that are given in search criteria.

```
<complexType name='CmNodes'>
```

```
    <complexContent>
        <restriction base='SOAP-ENC:Array'>
         <sequence>
          <element name='CmNode' type='tns:CmNode' minOccurs='0 maxOccurs='unbounded'/>
         </sequence>
        </restriction>
    </complexContent>
</complexType>
```

Each Unified CMNode contains a sequence of devices and their registration status.

```
    <complexType name='CmNode'>
        <sequence>
           <element name='ReturnCode'          type='tns:RisReturnCode'/>
           <element name='Name'                type='xsd:string'/>
           <element name='NoChange'            type='xsd:boolean'/>
           <element name='CmDevices'           type='tns:CmDevices'/>
         </sequence>
    </complexType>
    <complexType name='CmDevices'>
        <complexContent>
          <restriction base='SOAP-ENC:Array'>
          <sequence>
          <element name='CmDevice' type='tns:CmDevice' minOccurs='0' maxOccurs='200'/>
          </sequence>
          </restriction>
          </complexContent>
        </complexType>
```

The Unified CM Device information contains the following information.

```
<complexType name='CmDevice'>
<sequence>
<element name='Name' type='xsd:string'/>
<element name='IpAddress' type='xsd:string'/>
<element name='DirNumber' type='xsd:string'/>
<element name='Class' type='tns:DeviceClass'/>
<element name='Model' type='xsd:unsignedInt'/>
<element name='Product' type='xsd:unsignedInt'/>
<element name='BoxProduct'type='xsd:unsignedInt'/>
<element name='Httpd' type='tns:CmDevHttpd'/>
<element name='RegistrationAttempts' type='xsd:unsignedInt'/>
<element name='IsCtiControllable'  type='xsd:boolean'/>
<element name='LoginUserId' type='xsd:string'/>
<element name='Status'  type='tns:CmDevRegStat'/>
<element name='StatusReason' type='xsd:unsignedInt'/>
<element name='PerfMonObject' type='xsd:unsignedInt'/>
<element name='DChannel' type='xsd:unsignedInt'/>
<element name='Description' type='xsd:string'/>
<element name='H323Trunk' type='tns:H323Trunk'/>
<element name='TimeStamp' type='xsd:unsignedInt'/>
</sequence>
</complexType>
```

Selecting CTI Real-Time Information

Request Format

SOAP Action

This operation allows client to perform a CTI manager-related query.

The HTTP header should have following SOAP action:

```
SOAPAction: http://schemas.cisco.com/ast/soap/action/#RisPort#SelectCtiItems
```

Envelope Information

The query information should have an Envelope as follows:

```
1.  ?xml version="1.0" encoding="utf-8"?>
2.  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
           xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
3.  xmlns:tns="http://schemas.cisco.com/ast/soap/"
4.  xmlns:types="http://schemas.cisco.com/ast/soap/encodedTypes"
5.  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6.  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

Session ID

The following example shows a SOAP header that contains a session ID. The client sets a unique session ID.

```
7.  <soap:Header>
8.  <tns:AstHeader id="id1">
9.  <SessionId xsi:type="xsd:string">jSessionId</SessionId>
10. </tns:AstHeader>
11. </soap:Header>
12.
13. <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
14. <tns:SelectCtiItem><StateInfo xsi:type="xsd:string" /><CtiSelectionCriteria
href="#id1" /></tns:SelectCtiItem>
15. <tns:CtiSelectionCriteria id="id1" xsi:type="tns:CtiSelectionCriteria">
```

Maximum Device Information

The following example specifies the maximum number of devices that this search needs to return:

```
16. <MaxReturnedItems xsi:type="xsd:unsignedInt">10</MaxReturnedItems>
```

CTI Application/Device/Line Specification

The following example specifies on which CTI manager class Line/Device/Provider a search is provided:

```
17. <CtiMgrClass xsi:type="tns:CtiMgrClass">Line</CtiMgrClass>
```

Status of CTI Item Search

The following example specifies the Status of class on which to search:

```
18. <Status xsi:type="tns:CtiStatus">Any</Status>
```

Server Name for Search

The following example specifies the server name on which the search is performed:

```
19. <NodeName xsi:type="xsd:string" />
```

Type of Search

The following example specifies the type of selection:

```
20. <SelectAppBy xsi:type="tns:CtiSelectAppBy">AppIpAddress</SelectAppBy>
```

List of Items That Needs to be Searched

The following example specifies an array for items for which the real-time status is required:

```
21. <AppItems href="#id2" />Name /IP</tns:CtiSelectionCriteria>
22. <soapenc:Array id="id2" soapenc:arrayType="tns:SelectAppItem[2]">
23. <Item href="#id3" /><Item xsi:null="1" /></soapenc:Array>
24. <tns:SelectAppItem id="id3" xsi:type="tns:SelectAppItem">
25. <AppItem xsi:type="xsd:string"/>
26. </tns:SelectAppItem>
27. </soap:Body>
28. </soap:Envelope>
```

Response Format

The Response includes a sequence of Unified CM Nodes with sequences of CTI devices and lines real-time information:

```
<complexType name='CtiItem'>
<sequence>
<element name='AppId' type='xsd:string'/>
<element name='ProviderName' type='xsd:string'/>
<element name='UserId' type='xsd:string'/>
<element name='AppIpAddr' type='xsd:string'/>
<element name='AppStatus' type='tns:CtiStatus'/>
<element name='AppStatusReason' type='xsd:unsignedInt'/>
<element name='AppTimeStamp' type='xsd:unsignedInt'/>
<element name='CtiDevice' type='tns:CtiDevice'/>
<element name='CtiLine' type='tns:CtiLine'/>
</sequence>
</complexType>
```

CTI Device real-time information contains the following sequence of information:

```
<complexType name='CtiDevice'>
<sequence>
<element name='AppControlsMedia'        type='xsd:boolean'/>
<element name='DeviceName'              type='xsd:string'/>
<element name='DeviceStatus'            type='tns:CtiStatus'/>
<element name='DeviceStatusReason'      type='xsd:unsignedInt'/>
<element name='DeviceTimeStamp'         type='xsd:unsignedInt'/>
</sequence>
</complexType>
```

CTI Line contains the following sequence of information:

```
<complexType name='CtiLine'>
    <sequence>
        <element name='DirNumber'        type='xsd:string'/>
        <element name='LineStatus'       type='tns:CtiStatus'/>
        <element name='LineStatusReason' type='xsd:unsignedInt'/>
        <element name='LineTimeStamp'    type='xsd:unsignedInt'/>
     </sequence>
</complexType>
```

How to Get All Device Information in a Large System

The Serviceability-SOAP interface includes StateInfo as part of the response. This unique string tells the client the state of the Device server information. Clients must give this string from the first request to the next request for the same selection criteria. In that case, clients will get the response with the "NoChange" element set to True as part of CmNode. This means that the server information state has not changed from the previous state, which means that no new registration or deregistration of devices has happened. This saves lot of time for clients and servers because the server is telling the client that no change from the previous state has occurred. If "NoChange" in the response is set to False, this indicates the server information changed. In this case, the client needs to get the real-time information from the server and update the client information on the devices.

```
<!-- SOAP AST Header -->
<message name="AstHeader"><part name="AstHeader" type="tns:AstHeader"/></message>
<!-- R1. SelectCmDevice -->
<message name="SelectCmDeviceInput"><part name="StateInfo" type="xsd:string"/><part
name="CmSelectionCriteria" type="tns:CmSelectionCriteria"/>
</message>
<message name="SelectCmDeviceOutput"><part name="SelectCmDeviceResult"
type="tns:SelectCmDeviceResult"/><part name="StateInfo" type="xsd:string"/>
</message>
```

```
<complexType name="SelectCmDeviceResult">
<sequence><element name="TotalDevicesFound" type="xsd:unsignedInt"/><element
name="CmNodes" type="tns:CmNodes"/>
</sequence>
</complexType>
<complexType name="CmNodes">
<complexContent><restriction base="SOAP-ENC:Array"><attribute ref="soapenc:arrayType"
wsdl:arrayType="tns:CmNode[]"/>
</restriction>
</complexContent>
</complexType>
<complexType name="CmNode">
<sequence>
<element name="ReturnCode" type="tns:RisReturnCode"/><element name="Name"
type="xsd:string"/><element name="NoChange" type="xsd:boolean"/>
<element name="CmDevices" type="tns:CmDevices"/>
</sequence>
</complexType>
```

The system provides a maximum of 200 devices per response, as in the wsdl.

```
<complexType name="CmDevices"><complexContent>
<restriction base="SOAP-ENC:Array">
<sequence><element name="CmDevice" type="tns:CmDevice" minOccurs="0"
maxOccurs="200"/></sequence>
</restriction>
</complexContent>
</complexType>
```

The system does this to limit the response buffer to the first 200 devices per server even though the client may have given search criteria to get all the devices in a large deployment. Searches in call-processing servers are expensive in terms of CPU cycles. Call-processing servers need to service IP phone calls. The device requests from clients should consume less memory and CPU resources. Device requests cannot exceed 20 percent of CPU resources in call-processing servers. Many clients may be requesting this information, but Cisco has limited the information to the first 200 devices.

To get all configured device information, clients must use the AXL-DB API. Use the device information from the AXL-DB API in “SelectItems” of the “CmSelectionCriteria” Serviceability SOAP APIs to get real-time information for the first 200 devices.

```
<complexType name=" ">
<sequence><element name="MaxReturnedDevices" type="xsd:unsignedInt"/><element
name="Class" type="tns:DeviceClass"/><element name="Model"
type="xsd:unsignedInt"/><element name="Status" type="tns:CmDevRegStat"/><element
name="NodeName" type="xsd:string"/><element name="SelectBy"
type="tns:CmSelectBy"/><element name="SelectItems" type="tns:SelectItems"/>
</sequence>
</complexType>

<complexType name="SelectItems">
<complexContent>
<restriction base="SOAP-ENC:Array"><sequence><element name="SelectItem"
type="tns:SelectItem" minOccurs="0" maxOccurs="200"/></sequence>
</restriction>
</complexContent>
</complexType>
```

For the most efficient way to get the list of devices, use “Executesqlqueryreq()” of the AXL-DB API. One could use an SQL equivalent to “Select name from device where tkclass = 1” to get all phones, and then iterate through the list by sending 200 at a time to the AXIS-SOAP API.

Device requests per minute cannot exceed 15 per minute (by default) to the Cisco Unified Communications Manager server. Space client requests so they do not exceed 15 requests per minute. If the client requests exceed 15 per minute, the server will respond with a SOAP fault. Based on this fault, a SOAP client can adjust the request rate. The system expects these rates to take less than 20 percent of the CPU resources, which will not affect the ability of Cisco Unified Communications Manager to respond to IP Phone call requests.

Selecting SIP Device Real-Time Information

Request Format

SOAP Action

This operation allows clients to perform Cisco Unified Communications Manager SIP device related queries. The HTTP header contains the following SOAP action for these queries:

```
SOAPAction: "http://schemas.cisco.com/ast/soap/action/#RisPort#SelectCmDeviceSIP"
```

Envelope Information

Query information should have an Envelope as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
    <soapenv:Envelope xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
           <soapenv:Body>
                  <ns1:SelectCmDeviceSIP
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
```

State Info

If the same information is queried over and over again then Stateinfo needs to be sent from the previous request for each repetitive query by a client.

```
<StateInfo xsi:type="xsd:string"/>
```

The selection criteria type CmSelectionCriteriaSIP follows the optional State Info:

```
<CmSelectionCriteriaSIP href="#id0"/>
</ns1:SelectCmDeviceSIP>
```

CmSelectionCriteriaSIP comprises the following items:

- **MaxReturnedDevices**—Specifies how many maximum devices that can be returned for search criteria

  ```
  <MaxReturnedDevices xsi:type="xsd:unsignedInt">200</MaxReturnedDevices>
  ```

- **Class**—Specifies the device class type that needs to be queried for Real-time status. Device classes are Any, Phone, Gateway, H323, Cti, VoiceMail, MediaResources and Unknown.

  ```
  <Class xsi:type="xsd:string">Any</Class>
  ```

- **Model**—Specifies the model of the device. 255 applies for all models.

  ```
  <Model xsi:type="xsd:unsignedInt">255</Model>
  ```

- **Status**—Specifies the device status in search criteria, which is one of Any, Registered, UnRegistered, Rejected, PartiallyRegistered or Unknown.

```
<Status xsi:type="xsd:string">Registered</Status>
```

- **NodeName**—Specifies the server name where the search needs to be performed. If you do not specify a name, the system will search in all servers in cluster.

```
<NodeName xsi:type="xsd:string" xsi:nil="true"/>
```

- **SelectBy**—Specifies the selection type for whether it is IP Address/Name.

```
<SelectBy xsi:type="xsd:string">Name</SelectBy>
```

- **SelectItems**—Specifies the array of items for which search criteria is specified. The following specifies an array that contains the IP Address or Device Name of the items for which the real-time status is needed.

```
<SelectItems xsi:type="ns2:SelectItem" xsi:nil="true"/>
```

- **Protocol**—Specifies the protocol name in the search criteria, which is one of Any, SCCP, SIP, or Unknown.

```
<Protocol xsi:type="ns3:Protocol" xsi:nil="true"/>

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

Response Format

Response follows this schema and contains one to many node information, plus the stateInfo that the SOAP server returns. Each node includes a sequence of search information that was found based on the search criteria.

```
<complexType name='SelectCmDeviceResultSIP'>
            <sequence>
                   <element name='TotalDevicesFound' type='xsd:unsignedInt'/>
                   <element name='CmNodes' type='tns:CmNodesSIP'/>
            </sequence>
      </complexType>
```

CmNodesSIP is list of CmNodeSIP that are given in the search criteria.

```
<complexType name="CmNodesSIP">
            <complexContent>
                  <restriction base="SOAP-ENC:Array">
                        <attribute ref="soapenc:arrayType"
                     wsdl:arrayType="tns:CmNodeSIP[]"/>
                  </restriction>
            </complexContent>
       </complexType>
```

Each CmNodesSIP has a sequence of devices and their registration status.

```
    <complexType name='CmNodeSIP'>
        <sequence>
           <element name='ReturnCode'          type='tns:RisReturnCode'/>
           <element name='Name'                type='xsd:string'/>
           <element name='NoChange'            type='xsd:boolean'/>
           <element name='CmDevices'           type='tns:CmDevicesSIP'/>
         </sequence>
    </complexType>
        <complexType name="CmDevicesSIP">
             <complexContent>
```

```
                <restriction base="SOAP-ENC:Array">
                    <attribute ref="soapenc:arrayType"
                   wsdl:arrayType="tns:CmDeviceSIP[]"/>
                </restriction>
            </complexContent>
        </complexType>
```

CmDeviceSIP information will contain the following information:

```
<complexType name="CmDeviceSIP">
    <sequence>
        <element name="Name" type="xsd:string"/>
        <element name="IpAddress" type="xsd:string"/>
        <element name="DirNumber" type="xsd:string"/>
        <element name="Class" type="tns:DeviceClass"/>
        <element name="Model" type="xsd:unsignedInt"/>
        <element name="Product" type="xsd:unsignedInt"/>
        <element name="BoxProduct" type="xsd:unsignedInt"/>
        <element name="Httpd" type="tns:CmDevHttpd"/>
        <element name="RegistrationAttempts" type="xsd:unsignedInt"/>
        <element name="IsCtiControllable" type="xsd:boolean"/>
        <element name="LoginUserId" type="xsd:string"/>
        <element name="Status" type="tns:CmDevRegStat"/>
        <element name="StatusReason" type="xsd:unsignedInt"/>
        <element name="PerfMonObject" type="xsd:unsignedInt"/>
        <element name="DChannel" type="xsd:unsignedInt"/>
        <element name="Description" type="xsd:string"/>
        <element name="H323Trunk" type="tns:H323Trunk"/>
        <element name="TimeStamp" type="xsd:unsignedInt"/>
        <element name="Protocol" type="tns:ProtocolType"/>
        <element name="NumOfLines" type="xsd:unsignedInt"/>
        <element name="LinesStatus" type="tns:CmDevLinesStatus"/>
    </sequence>
</complexType>
```

Protocol defines the following enumerated protocol types:

```
    <simpleType name="ProtocolType">
            <restriction base="string">
                <enumeration value="Any"/>
                <enumeration value="SCCP"/>
                <enumeration value="SIP"/>
                <enumeration value="Unknown"/>
            </restriction>
        </simpleType>
```

CmDevLinesStatus is a list of CmDevSingleLineStatus:

```
    <complexType name="CmDevLinesStatus">
            <complexContent>
                <restriction base="SOAP-ENC:Array">
                    <attribute ref="soapenc:arrayType"
                    wsdl:arrayType="tns:CmDevSingleLineStatus[]"/>
                </restriction>
            </complexContent>
        </complexType>
```

CmSingleLineStatus is a sequence of DN and DN status:

```
        <complexType name="CmDevSingleLineStatus">
            <sequence>
                <element name="DirectoryNumber" type="xsd:string"/>
                <element name="Status" type="tns:CmSingleLineStatus"/>
            </sequence>
        </complexType>
```

CmSingleLineStatus defines the enumerated DN status as follows:

```
<simpleType name="CmSingleLineStatus">
    <restriction base="string">
        <enumeration value="Any"/>
        <enumeration value="Registered"/>
        <enumeration value="UnRegistered"/>
        <enumeration value="Rejected"/>
        <enumeration value="Unknown"/>
    </restriction>
</simpleType>
```

Example

The following example shows a SelectCmDeviceSIP response:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:SelectCmDeviceSIPResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <SelectCmDeviceResultSIP xsi:type="ns1:SelectCmDeviceResultSIP">
    <TotalDevicesFound xsi:type="xsd:unsignedInt">4</TotalDevicesFound>
    <CmNodes xsi:type="soapenc:Array" soapenc:arrayType="ns1:CmNodeSIP[2]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
     <item>
      <ReturnCode xsi:type="ns1:RisReturnCode">Ok</ReturnCode>
      <Name xsi:type="xsd:string">node70</Name>
      <NoChange xsi:type="xsd:boolean">false</NoChange>
      <CmDevices xsi:type="soapenc:Array" soapenc:arrayType="ns1:CmDeviceSIP[4]">
       <item>
        <Name xsi:type="xsd:string">SEP003094C25B01</Name>
        <IpAddress xsi:type="xsd:string">192.20.0.1</IpAddress>
        <DirNumber xsi:type="xsd:string">5001-Registered</DirNumber>
        <Class xsi:type="ns1:DeviceClass">Phone</Class>
        <Model xsi:type="xsd:unsignedInt">7</Model>
        <Product xsi:type="xsd:unsignedInt">35</Product>
        <BoxProduct xsi:type="xsd:unsignedInt" xsi:nil="true"/>
        <Httpd xsi:type="ns1:CmDevHttpd">Yes</Httpd>
        <RegistrationAttempts xsi:type="xsd:unsignedInt">0</RegistrationAttempts>
        <IsCtiControllable xsi:type="xsd:boolean">true</IsCtiControllable>
        <LoginUserId xsi:type="xsd:string">jdas0</LoginUserId>
        <Status xsi:type="ns1:CmDevRegStat">Registered</Status>
        <StatusReason xsi:type="xsd:unsignedInt">0</StatusReason>
        <PerfMonObject xsi:type="xsd:unsignedInt">2</PerfMonObject>
        <DChannel xsi:type="xsd:unsignedInt">0</DChannel>
        <Description xsi:type="xsd:string">Fake data</Description>
        <H323Trunk xsi:type="ns1:H323Trunk">
         <ConfigName xsi:type="xsd:string" xsi:nil="true"/>
         <TechPrefix xsi:type="xsd:string" xsi:nil="true"/>
         <Zone xsi:type="xsd:string" xsi:nil="true"/>
         <RemoteCmServer1 xsi:type="xsd:string" xsi:nil="true"/>
         <RemoteCmServer2 xsi:type="xsd:string" xsi:nil="true"/>
         <RemoteCmServer3 xsi:type="xsd:string" xsi:nil="true"/>
         <AltGkList xsi:type="xsd:string" xsi:nil="true"/>
         <ActiveGk xsi:type="xsd:string" xsi:nil="true"/>
         <CallSignalAddr xsi:type="xsd:string" xsi:nil="true"/>
         <RasAddr xsi:type="xsd:string" xsi:nil="true"/>
        </H323Trunk>
```

```
 <TimeStamp xsi:type="xsd:unsignedInt">1110841855</TimeStamp>
 <Protocol xsi:type="ns1:ProtocolType">SIP</Protocol>
 <NumOfLines xsi:type="xsd:unsignedInt">1</NumOfLines>
 <LinesStatus xsi:type="soapenc:Array"
        soapenc:arrayType="ns1:CmDevSingleLineStatus[1]">
  <item>
   <DirectoryNumber xsi:type="xsd:string">5001</DirectoryNumber>
   <Status xsi:type="ns1:CmSingleLineStatus">Registered</Status>
  </item>
 </LinesStatus>
</item>
<item>
 <Name xsi:type="xsd:string">SEP003094C25B02</Name>
 <IpAddress xsi:type="xsd:string">192.20.0.2</IpAddress>
 <DirNumber xsi:type="xsd:string">5002-Registered</DirNumber>
 <Class xsi:type="ns1:DeviceClass">Phone</Class>
 <Model xsi:type="xsd:unsignedInt">7</Model>
 <Product xsi:type="xsd:unsignedInt">35</Product>
 <BoxProduct xsi:type="xsd:unsignedInt" xsi:nil="true"/>
 <Httpd xsi:type="ns1:CmDevHttpd">Yes</Httpd>
 <RegistrationAttempts xsi:type="xsd:unsignedInt">0</RegistrationAttempts>
 <IsCtiControllable xsi:type="xsd:boolean">true</IsCtiControllable>
 <LoginUserId xsi:type="xsd:string">jdas1</LoginUserId>
 <Status xsi:type="ns1:CmDevRegStat">Registered</Status>
 <StatusReason xsi:type="xsd:unsignedInt">0</StatusReason>
 <PerfMonObject xsi:type="xsd:unsignedInt">2</PerfMonObject>
 <DChannel xsi:type="xsd:unsignedInt">0</DChannel>
 <Description xsi:type="xsd:string">Fake data</Description>
 <H323Trunk xsi:type="ns1:H323Trunk">
  <ConfigName xsi:type="xsd:string" xsi:nil="true"/>
  <TechPrefix xsi:type="xsd:string" xsi:nil="true"/>
  <Zone xsi:type="xsd:string" xsi:nil="true"/>
  <RemoteCmServer1 xsi:type="xsd:string" xsi:nil="true"/>
  <RemoteCmServer2 xsi:type="xsd:string" xsi:nil="true"/>
  <RemoteCmServer3 xsi:type="xsd:string" xsi:nil="true"/>
  <AltGkList xsi:type="xsd:string" xsi:nil="true"/>
       <ActiveGk xsi:type="xsd:string" xsi:nil="true"/>
  <CallSignalAddr xsi:type="xsd:string" xsi:nil="true"/>
  <RasAddr xsi:type="xsd:string" xsi:nil="true"/>
 </H323Trunk>
 <TimeStamp xsi:type="xsd:unsignedInt">1110841855</TimeStamp>
 <Protocol xsi:type="ns1:ProtocolType">SIP</Protocol>
 <NumOfLines xsi:type="xsd:unsignedInt">1</NumOfLines>
 <LinesStatus xsi:type="soapenc:Array"
        soapenc:arrayType="ns1:CmDevSingleLineStatus[1]">
  <item>
   <DirectoryNumber xsi:type="xsd:string">5002</DirectoryNumber>
   <Status xsi:type="ns1:CmSingleLineStatus">Registered</Status>
  </item>
 </LinesStatus>
</item>
<item>
 <Name xsi:type="xsd:string">SEP003094C25B03</Name>
 <IpAddress xsi:type="xsd:string">192.20.0.3</IpAddress>
 <DirNumber xsi:type="xsd:string">5003-Registered</DirNumber>
 <Class xsi:type="ns1:DeviceClass">Phone</Class>
 <Model xsi:type="xsd:unsignedInt">7</Model>
 <Product xsi:type="xsd:unsignedInt">35</Product>
 <BoxProduct xsi:type="xsd:unsignedInt" xsi:nil="true"/>
 <Httpd xsi:type="ns1:CmDevHttpd">Yes</Httpd>
 <RegistrationAttempts xsi:type="xsd:unsignedInt">0</RegistrationAttempts>
 <IsCtiControllable xsi:type="xsd:boolean">true</IsCtiControllable>
 <LoginUserId xsi:type="xsd:string">jdas2</LoginUserId>
 <Status xsi:type="ns1:CmDevRegStat">Registered</Status>
```

```
 <StatusReason xsi:type="xsd:unsignedInt">0</StatusReason>
 <PerfMonObject xsi:type="xsd:unsignedInt">2</PerfMonObject>
 <DChannel xsi:type="xsd:unsignedInt">0</DChannel>
 <Description xsi:type="xsd:string">Fake data</Description>
 <H323Trunk xsi:type="ns1:H323Trunk">
  <ConfigName xsi:type="xsd:string" xsi:nil="true"/>
  <TechPrefix xsi:type="xsd:string" xsi:nil="true"/>
  <Zone xsi:type="xsd:string" xsi:nil="true"/>
  <RemoteCmServer1 xsi:type="xsd:string" xsi:nil="true"/>
  <RemoteCmServer2 xsi:type="xsd:string" xsi:nil="true"/>
  <RemoteCmServer3 xsi:type="xsd:string" xsi:nil="true"/>
  <AltGkList xsi:type="xsd:string" xsi:nil="true"/>
  <ActiveGk xsi:type="xsd:string" xsi:nil="true"/>
  <CallSignalAddr xsi:type="xsd:string" xsi:nil="true"/>
  <RasAddr xsi:type="xsd:string" xsi:nil="true"/>
 </H323Trunk>
 <TimeStamp xsi:type="xsd:unsignedInt">1110841855</TimeStamp>
 <Protocol xsi:type="ns1:ProtocolType">SIP</Protocol>
 <NumOfLines xsi:type="xsd:unsignedInt">1</NumOfLines>
 <LinesStatus xsi:type="soapenc:Array"
        soapenc:arrayType="ns1:CmDevSingleLineStatus[1]">
  <item>
   <DirectoryNumber xsi:type="xsd:string">5003</DirectoryNumber>
   <Status xsi:type="ns1:CmSingleLineStatus">Registered</Status>
  </item>
 </LinesStatus>
</item>
<item>
 <Name xsi:type="xsd:string">SEP003094C25B04</Name>
 <IpAddress xsi:type="xsd:string">192.20.0.4</IpAddress>
 <DirNumber xsi:type="xsd:string">5004-Registered</DirNumber>
 <Class xsi:type="ns1:DeviceClass">Phone</Class>
 <Model xsi:type="xsd:unsignedInt">7</Model>
 <Product xsi:type="xsd:unsignedInt">35</Product>
 <BoxProduct xsi:type="xsd:unsignedInt" xsi:nil="true"/>
 <Httpd xsi:type="ns1:CmDevHttpd">Yes</Httpd>
 <RegistrationAttempts xsi:type="xsd:unsignedInt">0</RegistrationAttempts>
 <IsCtiControllable xsi:type="xsd:boolean">true</IsCtiControllable>
 <LoginUserId xsi:type="xsd:string">jdas3</LoginUserId>
 <Status xsi:type="ns1:CmDevRegStat">Registered</Status>
 <StatusReason xsi:type="xsd:unsignedInt">0</StatusReason>
 <PerfMonObject xsi:type="xsd:unsignedInt">2</PerfMonObject>
 <DChannel xsi:type="xsd:unsignedInt">0</DChannel>
 <Description xsi:type="xsd:string">Fake data</Description>
 <H323Trunk xsi:type="ns1:H323Trunk">
  <ConfigName xsi:type="xsd:string" xsi:nil="true"/>
  <TechPrefix xsi:type="xsd:string" xsi:nil="true"/>
  <Zone xsi:type="xsd:string" xsi:nil="true"/>
  <RemoteCmServer1 xsi:type="xsd:string" xsi:nil="true"/>
  <RemoteCmServer2 xsi:type="xsd:string" xsi:nil="true"/>
  <RemoteCmServer3 xsi:type="xsd:string" xsi:nil="true"/>
  <AltGkList xsi:type="xsd:string" xsi:nil="true"/>
  <ActiveGk xsi:type="xsd:string" xsi:nil="true"/>
  <CallSignalAddr xsi:type="xsd:string" xsi:nil="true"/>
  <RasAddr xsi:type="xsd:string" xsi:nil="true"/>
 </H323Trunk>
 <TimeStamp xsi:type="xsd:unsignedInt">1110841855</TimeStamp>
 <Protocol xsi:type="ns1:ProtocolType">SIP</Protocol>
 <NumOfLines xsi:type="xsd:unsignedInt">1</NumOfLines>
 <LinesStatus xsi:type="soapenc:Array"
           soapenc:arrayType="ns1:CmDevSingleLineStatus[1]">
  <item>
   <DirectoryNumber xsi:type="xsd:string">5004</DirectoryNumber>
   <Status xsi:type="ns1:CmSingleLineStatus">Registered</Status>
```

```
          </item>
         </LinesStatus>
        </item>
       </CmDevices>
      </item>
      <item>
       <ReturnCode xsi:type="ns1:RisReturnCode">NotFound</ReturnCode>
       <Name xsi:type="xsd:string">node71</Name>
       <NoChange xsi:type="xsd:boolean">false</NoChange>
       <CmDevices xsi:type="soapenc:Array" soapenc:arrayType="ns1:CmDeviceSIP[0]"/>
      </item>
     </CmNodes>
    </SelectCmDeviceResultSIP>
    <StateInfo xsi:type="xsd:string">&lt;StateInfo&gt;&lt;Node Name=&quot;node70&quot;
SubsystemStartTime=&quot;1110841841&quot; StateId=&quot;8&quot;
TotalItemsFound=&quot;4&quot; TotalItemsReturned=&quot;4&quot;/&gt;&lt;Node
Name=&quot;node71&quot; SubsystemStartTime=&quot;1110688669&quot; StateId=&quot;5772&quot;
TotalItemsFound=&quot;0&quot;
TotalItemsReturned=&quot;0&quot;/&gt;&lt;/StateInfo&gt;</StateInfo>
   </ns1:SelectCmDeviceSIPResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Interface to Get Server Names and Cluster Name

The interface to get cluster name getServiceParameter, interface to get configured servers in cluster listProcessNodeByService, and interface to get configured devices in cluster listDeviceByNameAndClass get defined as part of the AXL-DB WSDL file. Send your queries to API question mailer on these interfaces.

Authentication

The following sections describe the currently used authentication.

Note To improve performance, the SOAP server uses a timed finite cache of authentication and authorization information for up to 100 users. Cached information persists for up to 2 minutes.

Basic

Basic authentication uses base64 to encode and decode the credential information. Because base64 is a two-way function, which requires no key, this protocol represents an insecure clear-text authentication. Many platforms implement Basic authentication and most HTTP client agents support it. Basic authentication can be secure if encryption, such as SSL, is used.

Secure

When you install/upgrade Cisco Unified Communications Manager, the HTTPS self-signed certificate, httpscert.cer, automatically installs on the default website that hosts the Cisco Unified Communications Manager virtual directories.

Hypertext Transfer Protocol over Secure Sockets Layer (SSL), which secures communication between the browser client and the server, uses a certificate and a public key to encrypt the data that is transferred over the internet. HTTPS also ensures that the user login password transports securely via the web.

The following Cisco Unified Communications Manager applications support HTTPS, which ensures the identity of the server:

- Cisco Unified Communications Manager Administration
- Cisco Unified Communications Manager Serviceability
- Cisco Unified IP Phone User Option Pages
- Cisco Unified Communications Manager Bulk Administration
- Cisco Unified Communications Manager Auto-Register Phone Tool
- Cisco Unified Communications Manager CDR Analysis and Reporting
- Cisco Unified Communications Manager Trace Collection Tool
- Cisco Unified Communications Manager Real Time Monitoring Tool

For more information, refer to the *Cisco Unified Communications Manager Security Guide, Release 6.0*.

Authorization

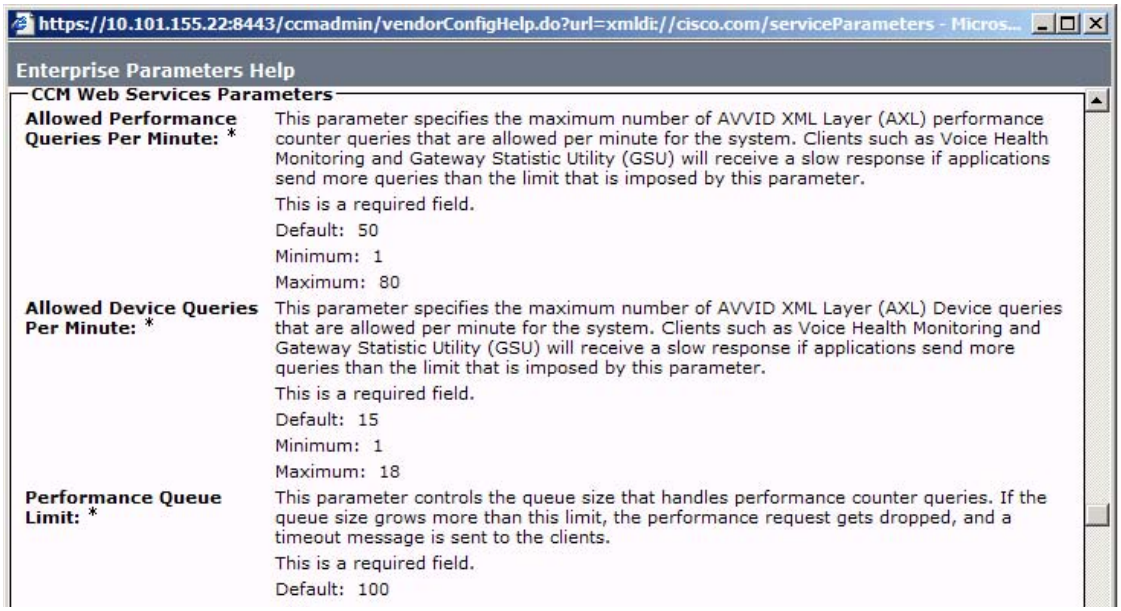
Each LDAP user gets checked against an MLA configuration for permissions. If the LDAP user in basic authentication does not have permission to "Read and Execute," the access to SOAP APIs gets denied.

Rate Control

The AXL-Serviceability Interface includes a request rate control mechanism that monitors the request rates for the complete system. If the request rate is more than supported, the system sends a “SOAP Fault” to the client and blocks the request. This rate control gets enabled for “Real-time Data requests” and “Perfmon Data Requests.” These rates get controlled through “Enterprise parameters” as shown in Figure 2-1. The system also provides Help for these parameter configurations as shown in Figure 2-2.

Figure 2-1 Rate Control Enterprise Parameters

Figure 2-2 Help for Rate Control Enterprise Parameters



Server Query Service

The system exports the following information from the Server Information SOAP interface:

- Host Name =MCS-SD4
- OS Name =Linux
- OS Arch =i386
- OS Version =2.4.21-15.ELsmp
- Java Runtime Version =1.4.2_05-b04
- Java Virtual Machine vendor =Sun Microsystems Inc.
- CallManager Version =6.0.1

The getServerInfo operation comprises the getServerInfoRequest and getServerInfoResponse:

```
 -<wsdl:operation name="getServerInfo">
  <wsdlsoap:operation
soapAction="http://schemas.cisco.com/ast/soap/action/#PerfmonPort#GetServerInfo" />
- <wsdl:input name="getServerInfoRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://schemas.cisco.com/ast/soap/" use="encoded" />
  </wsdl:input>
- <wsdl:output name="getServerInfoResponse">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://schemas.cisco.com/ast/soap/" use="encoded" />
  </wsdl:output>
  </wsdl:operation>
```

Request Format

In the request, an ArrayOfHosts definition gets specified, and the response provides the server information for the list of hostnames that are specified in the array.

```
- <wsdl:message name="getServerInfoRequest">
  <wsdl:part name="Hosts" type="impl:ArrayOfHosts" />
  </wsdl:message>
```

Response Format

The response comprises an ArrayOfServerInfo:

```
- <wsdl:message name="getServerInfoResponse">
  <wsdl:part name="ServerInfo" type="impl:ArrayOfServerInfo" />
  </wsdl:message>
- <complexType name="ArrayOfServerInfo">
- <complexContent>
- <restriction base="soapenc:Array">
  <attribute ref="soapenc:arrayType" wsdl:arrayType="impl:ServerInformation[]" />
  </restriction>
  </complexContent>
  </complexType>
```

ServerInformation comprises the following sequence of elements:

```
- <complexType name="ServerInformation">
- <sequence>
  <element name="HostName" nillable="true" type="xsd:string" />
  <element name="os-name" nillable="true" type="xsd:string" />
  <element name="os-version" nillable="true" type="xsd:string" />
  <element name="os-arch" nillable="true" type="xsd:string" />
  <element name="java-runtime-version" nillable="true" type="xsd:string" />
  <element name="java-vm-vendor" nillable="true" type="xsd:string" />
  <element name="call-manager-version" nillable="true" type="xsd:string" />
  </sequence>
  </complexType>
```

Faults

The Server will send a fault for “Error message context is NULL.” This error will not appear in normal operation.

The following fault gets sent if an HTTPS connection fails to a remote server — “Error initiating https connection to <URL>.”

Example

Request example

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:GetServerInfo soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <Hosts xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[1]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
    <item>172.19.240.90</item>
```

```
   </Hosts>
  </ns1:GetServerInfo>
 </soapenv:Body>
</soapenv:Envelope>
```

Response example

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:GetServerInfoResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <ServerInfo xsi:type="soapenc:Array" soapenc:arrayType="ns1:ServerInformation[1]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
    <item>
    <HostName xsi:type="xsd:string">MCS-SD4</HostName>
    <os-name xsi:type="xsd:string">Linux</os-name>
    <os-version xsi:type="xsd:string">2.4.21-15.ELsmp</os-version>
    <os-arch xsi:type="xsd:string">i386</os-arch>
    <java-runtime-version xsi:type="xsd:string">1.4.2_05-b04</java-runtime-version>
    <java-vm-vendor xsi:type="xsd:string">Sun Microsystems Inc.</java-vm-vendor>
    < Active_Versions xsi:type="xsd:string"> list of rpm separated by :
    </Active_Versions>
    <In_Active_Versions xsi:type="xsd:string">list of rpm separated by :
    </In_Active_Versions>
    </item>
   </ServerInfo>
  </ns1:GetServerInfoResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Service Interface

The Service Interface allows you to do Service Deploy, Service UnDeploy, Get Service List, and perform service starts and stops.

Get Static Service List

This **soapGetStaticServiceList** API allows clients to perform a query for all services static specification in the system. It returns the array of service static specification, which is composed of service name, type (servlet or service), deployable or undeployable service, group name, and dependent services.

Request Format

The HTTP header should have following SOAP action and envelop information:

```
<operation name="soapGetStaticServiceList">
<soap:operation
soapAction="http://schemas.cisco.com/ast/soap/action/#ControlCenterServices#soapGetSer
viceList"/>
    <input>
        <soap:body use="encoded" namespace="http://schemas.cisco.com/ast/soap/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output>
```

```
        <soap:body use="encoded" namespace="http://schemas.cisco.com/ast/soap/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
</operation>
```

The **soapGetStaticServiceList** operation takes only one dummy string typed parameter.

Response Format

The response contains the ArrayOfServiceSpecification information that is defined as follows:

```
<complexType name="StaticServiceList">
    <sequence>
        <element name="Services" type="tns:ArrayOfServiceSpecification"/>
    </sequence>
</complexType>
```

ArrayOfServiceSpecification is an array of ServiceSpecification defined as follows:

```
<complexType name="ArrayOfServiceSpecification">
    <complexContent>
        <restriction base="SOAP-ENC:Array">
        <attribute ref="SOAP-ENC:arrayType"
               wsdl:arrayType="tns:ServiceSpecification[]"/>
        </restriction>
    </complexContent>
</complexType>
```

ServiceSpecification contains information defined as follows:

```
<complexType name="ServiceSpecification">
    <sequence>
        <element name="ServiceName" type="xsd:string"/>
        <element name="ServiceType" type="tns:ServiceTypes"/>
        <element name="Deployable" type="boolean"/>
        <element name="GroupName" type="xsd:string"/>
        <element name="DependentServices" type="tns:ArrayOfServices"/>
    </sequence>
</complexType>
```

where ServiceTypes defines the type of service, defined as follows:

```
<simpleType name="ServiceTypes">
    <restriction base="xsd:string">
        <enumeration value="Service"/>
        <enumeration value="Servlet"/>
    </restriction>
</simpleType>
```

ArrayOfServices defines the dependent services for the service, defined as an array:

```
<complexType name="ArrayOfServices">
    <complexContent>
        <restriction base="SOAP-ENC:Array">
            <attribute ref="SOAP-ENC:arrayType" wsdl:arrayType="xsd:string[]"/>
        </restriction>
    </complexContent>
</complexType>
```

Fault

Invalid Service Configuration Files

Static service specification comes from two service configuration xml files:

- /usr/local/cm/conf/ccmservice/ActivateConf.xml
- /usr/local/cm/conf/ccmservice/ServicesConf.xml

If one of the files does not exist or has an invalid format, an empty list of static service specification will appear in the response.

Response Example

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:GetStaticServiceListResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <StaticServiceList xsi:type="ns2:StaticServiceList"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <Services xsi:type="soapenc:Array" soapenc:arrayType="ns2:ServiceSpecification[0]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
   </StaticServiceList>
  </ns1:GetStaticServiceListResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Request Example

The following example shows a request for getting a static service specification list :

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:GetStaticServiceList
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
      <ServiceInformationResponse
xsi:type="xsd:string">test</ServiceInformationResponse>
    </ns1:GetStaticServiceList>
  </soapenv:Body>
</soapenv:Envelope>
```

Response Example

The following example shows a response for getting a static service specification list:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:GetStaticServiceListResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
      <StaticServiceList xsi:type="ns2:StaticServiceList"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
```

```
        <Services xsi:type="soapenc:Array"
soapenc:arrayType="ns2:ServiceSpecification[50]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
          <item>
            <ServiceName xsi:type="xsd:string">Cisco Unified Communications
Manager</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <DependentServices xsi:type="soapenc:Array"
soapenc:arrayType="xsd:string[1]">
              <item>Cisco RIS Data Collector</item>
            </DependentServices>
          </item>
          <item>
            <ServiceName xsi:type="xsd:string">Cisco Tftp</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <DependentServices xsi:type="soapenc:Array"
soapenc:arrayType="xsd:string[1]">
              <item>Cisco RIS Data Collector</item>
            </DependentServices>
          </item>
          <item>
            <ServiceName xsi:type="xsd:string">Cisco Messaging Interface</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <DependentServices xsi:type="soapenc:Array"
soapenc:arrayType="xsd:string[1]">
              <item>Cisco RIS Data Collector</item>
            </DependentServices>
          </item>
          <item>
            <ServiceName xsi:type="xsd:string">Cisco IP Voice Media Streaming App
             </ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <DependentServices xsi:type="soapenc:Array"
                  soapenc:arrayType="xsd:string[1]">
              <item>Cisco RIS Data Collector</item>
            </DependentServices>
          </item>
          <item>
            <ServiceName xsi:type="xsd:string">Cisco CTIManager</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <DependentServices xsi:type="soapenc:Array"
                  soapenc:arrayType="xsd:string[2]">
              <item>Cisco RIS Data Collector</item>
              <item>Cisco Unified Communications Manager</item>
            </DependentServices>
          </item>
          <item>
            <ServiceName xsi:type="xsd:string">Cisco Unified Communications Manager
Attendant Console Server
             </ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CTI Services</GroupName>
            <DependentServices xsi:type="soapenc:Array"
                  soapenc:arrayType="xsd:string[1]">
              <item>Cisco RIS Data Collector</item>
            </DependentServices>
          </item>
          <item>
            <ServiceName xsi:type="xsd:string">Cisco RIS Data Collector</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
```

```
  <Deployable xsi:type="xsd:boolean">true</Deployable>
  <GroupName xsi:type="xsd:string">CM Services</GroupName>
  <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco Extension Mobility</ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
  <Deployable xsi:type="xsd:boolean">true</Deployable>
  <GroupName xsi:type="xsd:string">CM Services</GroupName>
  <DependentServices xsi:type="soapenc:Array"
      soapenc:arrayType="xsd:string[1]">
    <item>Cisco RIS Data Collector</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco Extended Functions</ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
  <Deployable xsi:type="xsd:boolean">true</Deployable>
  <GroupName xsi:type="xsd:string">Voice Quality Reporter Services
 </GroupName>
  <DependentServices xsi:type="soapenc:Array"
        soapenc:arrayType="xsd:string[1]">
    <item>Cisco RIS Data Collector</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco Serviceability Reporter
 </ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
  <Deployable xsi:type="xsd:boolean">true</Deployable>
  <GroupName xsi:type="xsd:string">CM Services</GroupName>
  <DependentServices xsi:type="soapenc:Array"
        soapenc:arrayType="xsd:string[2]">
    <item>Cisco RIS Data Collector</item>
    <item>Cisco AMC Service</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco CTL Provider</ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
  <Deployable xsi:type="xsd:boolean">true</Deployable>
  <GroupName xsi:type="xsd:string">Security Services</GroupName>
  <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco WebDialer Web Service
 </ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
  <Deployable xsi:type="xsd:boolean">true</Deployable>
  <GroupName xsi:type="xsd:string">CM Services</GroupName>
  <DependentServices xsi:type="soapenc:Array"
        soapenc:arrayType="xsd:string[1]">
    <item>Cisco RIS Data Collector</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco CDR Repository Manager
 </ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
  <Deployable xsi:type="xsd:boolean">true</Deployable>
  <GroupName xsi:type="xsd:string">CDR Services</GroupName>
  <DependentServices xsi:type="soapenc:Array"
        soapenc:arrayType="xsd:string[1]">
    <item>Cisco RIS Data Collector</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco Certificate Authority Proxy
     Function</ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
  <Deployable xsi:type="xsd:boolean">true</Deployable>
```

```
  <GroupName xsi:type="xsd:string">Security Services</GroupName>
  <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco CDR Agent</ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
  <Deployable xsi:type="xsd:boolean">true</Deployable>
  <GroupName xsi:type="xsd:string">CDR Services</GroupName>
  <DependentServices xsi:type="soapenc:Array"
         soapenc:arrayType="xsd:string[1]">
    <item>Cisco RIS Data Collector</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco SOAP - CDRonDemand Service
 </ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
  <Deployable xsi:type="xsd:boolean">true</Deployable>
  <GroupName xsi:type="xsd:string">CDR Services</GroupName>
  <DependentServices xsi:type="soapenc:Array"
         soapenc:arrayType="xsd:string[2]">
    <item>Cisco RIS Data Collector</item>
    <item>Cisco CDR Repository Manager</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco CAR Scheduler</ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
  <Deployable xsi:type="xsd:boolean">true</Deployable>
  <GroupName xsi:type="xsd:string">CDR Services</GroupName>
  <DependentServices xsi:type="soapenc:Array"
         soapenc:arrayType="xsd:string[3]">
    <item>Cisco RIS Data Collector</item>
    <item>Cisco CDR Repository Manager</item>
    <item>Cisco CAR Web Service</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco CAR Web Service</ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
  <Deployable xsi:type="xsd:boolean">true</Deployable>
  <GroupName xsi:type="xsd:string">CDR Services</GroupName>
  <DependentServices xsi:type="soapenc:Array"
         soapenc:arrayType="xsd:string[3]">
    <item>Cisco RIS Data Collector</item>
    <item>Cisco CDR Repository Manager</item>
    <item>Cisco CAR Scheduler</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco AMC Service</ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
  <Deployable xsi:type="xsd:boolean">true</Deployable>
  <GroupName xsi:type="xsd:string">CM Services</GroupName>
  <DependentServices xsi:type="soapenc:Array"
         soapenc:arrayType="xsd:string[1]">
    <item>Cisco RIS Data Collector</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco Unified CM SNMP Service<
     /ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
  <Deployable xsi:type="xsd:boolean">true</Deployable>
  <GroupName xsi:type="xsd:string">CM Services</GroupName>
  <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco DirSync</ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
  <Deployable xsi:type="xsd:boolean">true</Deployable>
```

```
                    <GroupName xsi:type="xsd:string">CM Services</GroupName>
                    <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
                  </item>
                  <item>
                    <ServiceName xsi:type="xsd:string">Cisco AXL Web Service</ServiceName>
                    <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
                    <Deployable xsi:type="xsd:boolean">true</Deployable>
                    <GroupName xsi:type="xsd:string">Database and Admin Services</GroupName>
                    <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
                  </item>
                  <item>
                    <ServiceName xsi:type="xsd:string">Cisco DRF Master</ServiceName>
                    <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
                    <Deployable xsi:type="xsd:boolean">true</Deployable>
                    <GroupName xsi:type="xsd:string">CM Services</GroupName>
                    <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
                  </item>
                  <item>
                    <ServiceName xsi:type="xsd:string">Cisco DRF Local</ServiceName>
                    <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
                    <Deployable xsi:type="xsd:boolean">true</Deployable>
                    <GroupName xsi:type="xsd:string">CM Services</GroupName>
                    <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
                  </item>
                  <item>
                    <ServiceName xsi:type="xsd:string">Cisco Unified IP Phone Services<
                       /ServiceName>
                    <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
                    <Deployable xsi:type="xsd:boolean">true</Deployable>
                    <GroupName xsi:type="xsd:string">CM Services</GroupName>
                    <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
                  </item>
                  <item>
                    <ServiceName xsi:type="xsd:string">Cisco Unified Communications
Manager</ServiceName>
                    <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
                    <Deployable xsi:type="xsd:boolean">false</Deployable>
                    <GroupName xsi:type="xsd:string">CM Services</GroupName>
                    <DependentServices xsi:type="soapenc:Array"
                          soapenc:arrayType="xsd:string[1]">
                      <item>Cisco RIS Data Collector</item>
                    </DependentServices>
                  </item>
                  <item>
                    <ServiceName xsi:type="xsd:string">Cisco Tftp</ServiceName>
                    <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
                    <Deployable xsi:type="xsd:boolean">false</Deployable>
                    <GroupName xsi:type="xsd:string">CM Services</GroupName>
                    <DependentServices xsi:type="soapenc:Array"
                          soapenc:arrayType="xsd:string[1]">
                      <item>Cisco RIS Data Collector</item>
                    </DependentServices>
                  </item>
                  <item>
                    <ServiceName xsi:type="xsd:string">Cisco Messaging Interface</ServiceName>
                    <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
                    <Deployable xsi:type="xsd:boolean">false</Deployable>
                    <GroupName xsi:type="xsd:string">CM Services</GroupName>
                    <DependentServices xsi:type="soapenc:Array"
                          soapenc:arrayType="xsd:string[1]">
                      <item>Cisco RIS Data Collector</item>
                    </DependentServices>
                  </item>
                  <item>
                    <ServiceName xsi:type="xsd:string">Cisco IP Voice Media Streaming App
                   </ServiceName>
                    <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
                    <Deployable xsi:type="xsd:boolean">false</Deployable>
                    <GroupName xsi:type="xsd:string">CM Services</GroupName>
                    <DependentServices xsi:type="soapenc:Array"
                          soapenc:arrayType="xsd:string[1]">
```

```
                        <item>Cisco RIS Data Collector</item>
                      </DependentServices>
                    </item>
                    <item>
                      <ServiceName xsi:type="xsd:string">Cisco CTIManager</ServiceName>
                      <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
                      <Deployable xsi:type="xsd:boolean">false</Deployable>
                      <GroupName xsi:type="xsd:string">CM Services</GroupName>
                      <DependentServices xsi:type="soapenc:Array"
                             soapenc:arrayType="xsd:string[2]">
                        <item>Cisco RIS Data Collector</item>
                        <item>Cisco Unified Communications Manager</item>
                      </DependentServices>
                    </item>
                    <item>
                      <ServiceName xsi:type="xsd:string">Cisco Unified Communications Manager
Attendant
                         Console Server</ServiceName>
                      <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
                      <Deployable xsi:type="xsd:boolean">false</Deployable>
                      <GroupName xsi:type="xsd:string">CTI Services</GroupName>
                      <DependentServices xsi:type="soapenc:Array"
                             soapenc:arrayType="xsd:string[1]">
                        <item>Cisco RIS Data Collector</item>
                      </DependentServices>
                    </item>
                    <item>
                      <ServiceName xsi:type="xsd:string">Cisco RIS Data Collector</ServiceName>
                      <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
                      <Deployable xsi:type="xsd:boolean">false</Deployable>
                      <GroupName xsi:type="xsd:string">CM Services</GroupName>
                      <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
                    </item>
                    <item>
                      <ServiceName xsi:type="xsd:string">Cisco Extension Mobility</ServiceName>
                      <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
                      <Deployable xsi:type="xsd:boolean">false</Deployable>
                      <GroupName xsi:type="xsd:string">CM Services</GroupName>
                      <DependentServices xsi:type="soapenc:Array"
                             soapenc:arrayType="xsd:string[1]">
                    <item>Cisco RIS Data Collector</item>
                      </DependentServices>
                    </item>
                    <item>
                      <ServiceName xsi:type="xsd:string">Cisco Extended Functions</ServiceName>
                      <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
                      <Deployable xsi:type="xsd:boolean">false</Deployable>
                      <GroupName xsi:type="xsd:string">Voice Quality Reporter Services
                     </GroupName>
                      <DependentServices xsi:type="soapenc:Array"
                             soapenc:arrayType="xsd:string[1]">
                        <item>Cisco RIS Data Collector</item>
                      </DependentServices>
                    </item>
                    <item>
                      <ServiceName xsi:type="xsd:string">Cisco Serviceability Reporter
                     </ServiceName>
                      <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
                      <Deployable xsi:type="xsd:boolean">false</Deployable>
                      <GroupName xsi:type="xsd:string">CM Services</GroupName>
                      <DependentServices xsi:type="soapenc:Array"
                         soapenc:arrayType="xsd:string[2]">
                        <item>Cisco RIS Data Collector</item>
                        <item>Cisco AMC Service</item>
                      </DependentServices>
                    </item>
                    <item>
                      <ServiceName xsi:type="xsd:string">Cisco CTL Provider</ServiceName>
                      <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
                      <Deployable xsi:type="xsd:boolean">false</Deployable>
                      <GroupName xsi:type="xsd:string">Security Services</GroupName>
```

```
  <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco WebDialer Web Service
 </ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
  <Deployable xsi:type="xsd:boolean">false</Deployable>
  <GroupName xsi:type="xsd:string">CM Services</GroupName>
  <DependentServices xsi:type="soapenc:Array"
        soapenc:arrayType="xsd:string[1]">
    <item>Cisco RIS Data Collector</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco CDR Repository Manager
 </ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
  <Deployable xsi:type="xsd:boolean">false</Deployable>
  <GroupName xsi:type="xsd:string">CDR Services</GroupName>
  <DependentServices xsi:type="soapenc:Array"
        soapenc:arrayType="xsd:string[1]">
    <item>Cisco RIS Data Collector</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco Certificate Authority Proxy
     Function</ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
  <Deployable xsi:type="xsd:boolean">false</Deployable>
  <GroupName xsi:type="xsd:string">Security Services</GroupName>
  <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco CDR Agent</ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
  <Deployable xsi:type="xsd:boolean">false</Deployable>
  <GroupName xsi:type="xsd:string">CDR Services</GroupName>
  <DependentServices xsi:type="soapenc:Array"
        soapenc:arrayType="xsd:string[1]">
    <item>Cisco RIS Data Collector</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco SOAP - CDROnDemand Service
 </ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
  <Deployable xsi:type="xsd:boolean">false</Deployable>
  <GroupName xsi:type="xsd:string">CDR Services</GroupName>
  <DependentServices xsi:type="soapenc:Array"
        soapenc:arrayType="xsd:string[2]">
    <item>Cisco RIS Data Collector</item>
    <item>Cisco CDR Repository Manager</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco CAR Scheduler</ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
  <Deployable xsi:type="xsd:boolean">false</Deployable>
  <GroupName xsi:type="xsd:string">CDR Services</GroupName>
  <DependentServices xsi:type="soapenc:Array"
        soapenc:arrayType="xsd:string[3]">
    <item>Cisco RIS Data Collector</item>
    <item>Cisco CDR Repository Manager</item>
    <item>Cisco CAR Web Service</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco CAR Web Service</ServiceName>
  <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
  <Deployable xsi:type="xsd:boolean">false</Deployable>
  <GroupName xsi:type="xsd:string">CDR Services</GroupName>
```

```
            <DependentServices xsi:type="soapenc:Array"
                   soapenc:arrayType="xsd:string[3]">
              <item>Cisco RIS Data Collector</item>
              <item>Cisco CDR Repository Manager</item>
              <item>Cisco CAR Scheduler</item>
            </DependentServices>
          </item>
          <item>
            <ServiceName xsi:type="xsd:string">Cisco AMC Service</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <DependentServices xsi:type="soapenc:Array"
                   soapenc:arrayType="xsd:string[1]">
              <item>Cisco RIS Data Collector</item>
            </DependentServices>
          </item>
          <item>
            <ServiceName xsi:type="xsd:string">Cisco Unified CM SNMP Service<
               /ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item>
            <ServiceName xsi:type="xsd:string">Cisco DirSync</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item>
            <ServiceName xsi:type="xsd:string">Cisco AXL Web Service</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">Database and Admin Services</GroupName>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item>
            <ServiceName xsi:type="xsd:string">Cisco DRF Master</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item>
            <ServiceName xsi:type="xsd:string">Cisco DRF Local</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item>
            <ServiceName xsi:type="xsd:string">Cisco Unified IP Phone Services<
               /ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
        </Services>
      </StaticServiceList>
    </ns1:GetStaticServiceListResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Get Service Status API

This **soapGetServiceStatus** API allows clients to perform a list of deployable/undeployable services status query. It returns the service status response information for the services that have been queried. It also allows getting all of the services current status by providing an empty list of services.

Request Format

HTTP header should have following SOAP action and envelop information:

```
<operation name="soapGetServiceStatus">
    <soap:operation
soapAction="http://schemas.cisco.com/ast/soap/action/#ControlCenterServices#soapGetSer
viceStatus"/>
    <input>
        <soap:body use="encoded" namespace="http://schemas.cisco.com/ast/soap/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output>
        <soap:body use="encoded" namespace="http://schemas.cisco.com/ast/soap/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
</operation>
```

The **soapGetServiceStatus** operation takes one array of service names for which their statuses are queried.

```
<GetServiceStatusList xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[1]"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
        <item>service name</item>
</GetServiceStatusList>
```

If the array of service names is empty in the request, the response will contain service status information for all services in the system.

Response Format

The response contains the performed query information that is defined as follows:

```
<complexType name="ServiceInformationResponse">
    <sequence>
        <element name="ReturnCode" type="tns:ReturnCode"/>
        <element name="ReasonCode" type="xsd:integer"/>
        <element name="ReasonString" type="xsd:string"/>
        <element name="ServiceInfoList" type="tns:ArrayOfServiceInformation"/>
    </sequence>
</complexType>
```

where ReturnCode will be a string format of return code:

```
<simpleType name="ReturnCode">
    <restriction base="xsd:string"/>
</simpleType>
```

ArrayOfServiceInformation is an array of ServiceInformation that defines service name, service status, reason code, reason string, service start time, and service up time.

```
<complexType name="ArrayOfServiceInformation">
    <complexContent>
    <restriction base="SOAP-ENC:Array">
```

```
        <attribute ref="SOAP-ENC:arrayType" wsdl:arrayType="tns:ServiceInformation[
]"/>
    </restriction>
</complexContent>
</complexType>
<complexType name="ServiceInformation">
    <sequence>
        <element name="ServiceName" type="xsd:string"/>
        <element name="ServiceStatus" type="tns:ServiceStatus"/>
        <element name="ReasonCode" type="xsd:integer"/>
        <element name="ReasonCodeString" type="xsd:string"/>
        <element name="StartTime" type="xsd:string"/>
        <element name="UpTime" type="xsd:integer"/>
    </sequence>
</complexType>
```

ServiceStatus defines the enumerated status for the service, as follows:

```
<simpleType name="ServiceStatus">
    <restriction base="xsd:string">
        <enumeration value="Started"/>
        <enumeration value="Stopped"/>
        <enumeration value="Starting"/>
        <enumeration value="Stopping"/>
        <enumeration value="Unknown"/>
    </restriction>
</simpleType>
```

The following details apply about ServiceStatus, ReasonCode, and ReasonCodeString.

- ServiceStatus is either Started or Stopped. If Started, StartTime gives the time that the service is started in a time string format; UpTime gives the time in seconds since the service was started.
- The ReasonCode and ReasonCodeString explain the reason that the service is Stopped:
  - If a deployable service is activated and stopped by admin, its status is Stopped, the ReasonCode equals -1019, and the corresponding ReasonCodeString specifies “Component is not running”.
  - If a deployable service is deactivated, its status is Stopped, the ReasonCode equals -1068, and the corresponding ReasonCodeString specifies “Service Not Activated”.
  - If a nondeployable item is stopped by admin, its status could be Stopped with ReasonCode -1019, and the corresponding ReasonCodeString “Component is not running.”

    The complete listing of ReasonCode and ReasonCodeString follows:

```
-1000: "Component already initialized"
-1001: "Entry replaced"
-1002: "Component not initialized"
-1003: "Component is running"
-1004: "Entry not found"
-1005: "Unable to process event"
-1006: "Registration already present"
-1007: "Unsuccessful completion"
-1008: "Registration not found"
-1009: "Missing or invalid environment variable"
-1010: "No such service"
-1011: "Component is reserved for platform"
-1012: "Bad arguments"
-1013: "Internal error"
-1014: "Entry was already present"
-1015: "Error opening IPC"
-1016: "No license available"
-1017: "Error opening file"
-1018: "Error reading file"
-1019: "Component is not running"
-1020: "Signal ignored"
-1021: "Notification ignored"
```

```
-1022: "Buffer overflow"
-1023: "Cannot parse record or entry"
-1024: "Out of memory"
-1025: "Not connected"
-1026: "Component already exists"
-1027: "Message was truncated"
-1028: "Component is empty"
-1029: "Operation is pending"
-1030: "Transaction does not exist"
-1031: "Operation timed-out"
-1032: "File is locked"
-1033: "Feature is not implemented yet"
-1034: "Alarm was already set"
-1035: "Alarm was already clear"
-1036: "Dependency is in active state"
-1037: "Dependency is not in active state"
-1038: "Circular dependencies detected"
-1039: "Component already started"
-1040: "Component already stopped"
-1041: "Dependencies still pending"
-1042: "Requested process state transition not allowed"
-1043: "No changes"
-1044: "Boundary violation for data structure"
-1045: "Operation not supported"
-1046: "Process recovery in progress"
-1047: "Operation pending on scheduled restart"
-1048: "Operation pending on active dependencies"
-1049: "Operation pending on active dependents"
-1050: "Shutdown is in progress"
-1051: "Invalid Table Handle"
-1052: "Data Base not initialized"
-1053: "Data Directory"
-1054: "Table Full"
-1055: "Deleted Data"
-1056: "No Such Record"
-1057: "Component already in specified state"
-1058: "Out of range"
-1059: "Cannot create object"
-1060: "MSO refused, standby system not ready."
-1061: "MSO refused, standby state update still in progress. Try again later."
-1062: "MSO refused, standby state update failed.
           Verify configuration on standby."
-1063: "MSO refused, Warm start-up in progress."
-1064: "MSO refused, Warm start-up Failed."
-1065: "MSO refused, System is not in active state"
-1066: "MSO refused, Detected standalone Flag"
-1067: "Invalid Token presented in reques"
-1068: "Service Not Activated"
-1069: "Commanded Out of Service"
-1070: "Multiple Modules have error"
-1071: "Encountered exception"
-1072: "Invalid context path was specified"
-1073: "No context exists"
-1074: "No context path was specified"
-1075: "Application already exists"
```

Fault

Invalid Service: Non-existing Service

If the request for getting the service status is for an invalid service name, such as a nonexistent service name, the ReasonCode -1010 and the ReasonCodeString “No such service” will appear in the response. The following request and response example applies for getting service status for “Cisco WrongService.”

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:GetServiceStatus
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <GetServiceStatusList xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[1]"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
    <item>Cisco WrongService</item>
   </GetServiceStatusList>
  </ns1:GetServiceStatus>
 </soapenv:Body>
</soapenv:Envelope>

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:GetServiceStatusResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <ServiceInformationResponse xsi:type="ns2:ServiceInformationResponse"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <ReturnCode xsi:type="ns2:ReturnCode">0</ReturnCode>
    <ReasonCode xsi:type="xsd:integer">-1010</ReasonCode>
    <ReasonString xsi:type="xsd:string">No such service</ReasonString>
    <ServiceInfoList xsi:type="ns2:ServiceInformation" xsi:nil="true"/>
   </ServiceInformationResponse>
  </ns1:GetServiceStatusResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Request Example

The following request example for getting “Cisco Tftp” service status applies:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:GetServiceStatus
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
      <GetServiceStatusList xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[1]"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
        <item>Cisco Tftp</item>
      </GetServiceStatusList>
    </ns1:GetServiceStatus>
  </soapenv:Body>
</soapenv:Envelope>
```

Response Example

The following response example for getting “Cisco Tftp” service status applies:

```
 <?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:GetServiceStatusResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
      <ServiceInformationResponse xsi:type="ns2:ServiceInformationResponse"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
        <ReturnCode xsi:type="ns2:ReturnCode">0</ReturnCode>
        <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
        <ReasonString xsi:type="xsd:string" xsi:nil="true"/>
        <ServiceInfoList xsi:type="soapenc:Array"
soapenc:arrayType="ns2:ServiceInformation[1]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
          <item>
            <ServiceName xsi:type="xsd:string">Cisco Tftp</ServiceName>
            <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
            <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
            <ReasonCodeString xsi:type="xsd:string">
            </ReasonCodeString>
            <StartTime xsi:type="xsd:string">Tue Sep 14 14:29:43 2004</StartTime>
            <UpTime xsi:type="xsd:integer">11800</UpTime>
          </item>
        </ServiceInfoList>
      </ServiceInformationResponse>
    </ns1:GetServiceStatusResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Request Example

The following request example for getting service status when providing an empty array of service names applies:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:GetServiceStatus
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <GetServiceStatusList xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[0]"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
  </ns1:GetServiceStatus>
 </soapenv:Body>
</soapenv:Envelope>
```

Response Example

The response for the preceding request gets the current status for all services as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:GetServiceStatusResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <ServiceInformationResponse xsi:type="ns2:ServiceInformationResponse"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <ReturnCode xsi:type="ns2:ReturnCode">0</ReturnCode>
    <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
    <ReasonString xsi:type="xsd:string" xsi:nil="true"/>
    <ServiceInfoList xsi:type="soapenc:Array"
soapenc:arrayType="ns2:ServiceInformation[43]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
     <item>
      <ServiceName xsi:type="xsd:string">A Red Hat DB</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:14 2004</StartTime>
      <UpTime xsi:type="xsd:integer">186704</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco AMC Service</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Mon Dec  6 17:51:26 2004</StartTime>
      <UpTime xsi:type="xsd:integer">161372</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco AXL Web Service</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Tue Dec  7 18:13:04 2004</StartTime>
      <UpTime xsi:type="xsd:integer">73674</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">CiscoUnified CM SNMP Service</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:19 2004</StartTime>
      <UpTime xsi:type="xsd:integer">186699</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco CDP</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:19 2004</StartTime>
      <UpTime xsi:type="xsd:integer">186699</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco CDP Agent</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:19 2004</StartTime>
      <UpTime xsi:type="xsd:integer">186699</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco CDR Agent</ServiceName>
```

```
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Mon Dec  6 10:50:38 2004</StartTime>
      <UpTime xsi:type="xsd:integer">186620</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco CTIManager</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:20 2004</StartTime>
      <UpTime xsi:type="xsd:integer">186698</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco CTL Provider</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:20 2004</StartTime>
      <UpTime xsi:type="xsd:integer">186698</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco Unified Communications
Manager</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Mon Dec  6 12:43:07 2004</StartTime>
      <UpTime xsi:type="xsd:integer">179871</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco Unified Communications Manager
Admin</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Tue Dec  7 18:13:01 2004</StartTime>
      <UpTime xsi:type="xsd:integer">73677</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco Unified Communications Manager Attendant
Console
           Server</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:20 2004</StartTime>
      <UpTime xsi:type="xsd:integer">186698</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco Unified Communications Manager Personal
Directory
       </ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Tue Dec  7 18:13:02 2004</StartTime>
      <UpTime xsi:type="xsd:integer">73676</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco Unified Communications Manager
Serviceability<
           /ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Tue Dec  7 18:13:13 2004</StartTime>
      <UpTime xsi:type="xsd:integer">73665</UpTime>
     </item>
     <item>
```

```
      <ServiceName xsi:type="xsd:string">Cisco Unified Communications Manager
Serviceability RTMT
       </ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Tue Dec  7 18:13:03 2004</StartTime>
      <UpTime xsi:type="xsd:integer">73675</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco DRF Local</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Stopped</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1019</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string">Component is not running</ReasonCodeString>
      <StartTime xsi:type="xsd:string" xsi:nil="true"/>
      <UpTime xsi:type="xsd:integer">-1</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco Database Layer Monitor</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:20 2004</StartTime>
      <UpTime xsi:type="xsd:integer">186698</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco DirSync</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:20 2004</StartTime>
      <UpTime xsi:type="xsd:integer">186698</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco Electronic Notification</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:20 2004</StartTime>
      <UpTime xsi:type="xsd:integer">186698</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco Extended Functions</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:20 2004</StartTime>
      <UpTime xsi:type="xsd:integer">186698</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco Extension Mobility</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Tue Dec  7 18:12:57 2004</StartTime>
      <UpTime xsi:type="xsd:integer">73681</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco Extension Mobility Application
       </ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Tue Dec  7 18:12:39 2004</StartTime>
      <UpTime xsi:type="xsd:integer">73699</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco IP Voice Media Streaming App</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
```

```
 <StartTime xsi:type="xsd:string">Mon Dec  6 12:30:51 2004</StartTime>
 <UpTime xsi:type="xsd:integer">180607</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Cisco Log Partition Monitoring Tool</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
 <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:20 2004</StartTime>
 <UpTime xsi:type="xsd:integer">186698</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Cisco Messaging Interface</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Stopped</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1019</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string">Component is not running</ReasonCodeString>
 <StartTime xsi:type="xsd:string" xsi:nil="true"/>
 <UpTime xsi:type="xsd:integer">-1</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Cisco RIS Data Collector</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
 <StartTime xsi:type="xsd:string">Mon Dec  6 16:25:25 2004</StartTime>
 <UpTime xsi:type="xsd:integer">166533</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Cisco RTMT Reporter Servlet</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
 <StartTime xsi:type="xsd:string">Tue Dec  7 18:12:56 2004</StartTime>
 <UpTime xsi:type="xsd:integer">73682</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Cisco SOAP -Log Collection APIs</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
 <StartTime xsi:type="xsd:string">Tue Dec  7 18:12:56 2004</StartTime>
 <UpTime xsi:type="xsd:integer">73682</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Cisco SOAP - Performance Monitoring APIs
   </ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
 <StartTime xsi:type="xsd:string">Tue Dec  7 18:12:58 2004</StartTime>
 <UpTime xsi:type="xsd:integer">73680</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Cisco SOAP -Real-Time Service APIs</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
 <StartTime xsi:type="xsd:string">Tue Dec  7 18:12:59 2004</StartTime>
 <UpTime xsi:type="xsd:integer">73679</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Cisco Serviceability Reporter</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
 <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:21 2004</StartTime>
 <UpTime xsi:type="xsd:integer">186697</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Cisco Syslog Agent</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
```

```
 <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
 <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:21 2004</StartTime>
 <UpTime xsi:type="xsd:integer">186697</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Cisco Tftp</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
 <StartTime xsi:type="xsd:string">Mon Dec  6 17:51:20 2004</StartTime>
 <UpTime xsi:type="xsd:integer">161378</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Cisco Tomcat</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
 <StartTime xsi:type="xsd:string">Tue Dec  7 18:12:35 2004</StartTime>
 <UpTime xsi:type="xsd:integer">73703</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Cisco WebDialer Web Service</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
 <StartTime xsi:type="xsd:string">Tue Dec  7 18:13:00 2004</StartTime>
 <UpTime xsi:type="xsd:integer">73678</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Host Resources Agent</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
 <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:56 2004</StartTime>
 <UpTime xsi:type="xsd:integer">186662</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">MIB2 Agent</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
 <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:58 2004</StartTime>
 <UpTime xsi:type="xsd:integer">186660</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Native Agent Adaptor</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
 <StartTime xsi:type="xsd:string">Mon Dec  6 10:49:59 2004</StartTime>
 <UpTime xsi:type="xsd:integer">186659</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">SNMP Master Agent</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Stopped</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1019</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string">Component is not running</ReasonCodeString>
 <StartTime xsi:type="xsd:string" xsi:nil="true"/>
 <UpTime xsi:type="xsd:integer">-1</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Cisco CAR Scheduler</ServiceName>
 <ServiceStatus xsi:type="ns2:ServiceStatus">Stopped</ServiceStatus>
 <ReasonCode xsi:type="xsd:integer">-1068</ReasonCode>
 <ReasonCodeString xsi:type="xsd:string">Service Not Activated </ReasonCodeString>
 <StartTime xsi:type="xsd:string" xsi:nil="true"/>
 <UpTime xsi:type="xsd:integer">-1</UpTime>
</item>
<item>
 <ServiceName xsi:type="xsd:string">Cisco CAR Web Service</ServiceName>
```

```
      <ServiceStatus xsi:type="ns2:ServiceStatus">Stopped</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1068</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string">Service Not Activated </ReasonCodeString>
      <StartTime xsi:type="xsd:string" xsi:nil="true"/>
      <UpTime xsi:type="xsd:integer">-1</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco CDR Repository Manager</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Stopped</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1068</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string">Service Not Activated </ReasonCodeString>
      <StartTime xsi:type="xsd:string" xsi:nil="true"/>
      <UpTime xsi:type="xsd:integer">-1</UpTime>
     </item>
     <item>
      <ServiceName xsi:type="xsd:string">Cisco SOAP - CDRonDemand Service</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Stopped</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1068</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string">Service Not Activated </ReasonCodeString>
      <StartTime xsi:type="xsd:string" xsi:nil="true"/>
      <UpTime xsi:type="xsd:integer">-1</UpTime>
     </item>
    </ServiceInfoList>
   </ServiceInformationResponse>
  </ns1:GetServiceStatusResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Do Service Deployment API

This **soapDoServiceDeployment** API allows clients to deploy or undeploy a list of services. It returns the services response information for the services that are requested to be deployed or undeployed. You can use this API only to deploy or undeploy a deployable service, a service with the Deployable attribute set to True in the response from getting the static service specification. The API does not allow clients to deploy or undeploy an empty list of services.

Request Format

The HTTP header should have following SOAP action and envelop information:

```
<operation name="soapDoServiceDeployment">
    <soap:operation
soapAction="http://schemas.cisco.com/ast/soap/action/#ControlCenterServices#soapDoServ
iceDeployment"/>
    <input>
        <soap:body use="encoded" namespace="http://schemas.cisco.com/ast/soap/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output>
        <soap:body use="encoded" namespace="http://schemas.cisco.com/ast/soap/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
</operation>
```

The **soapDoServiceDeployment** operation takes one array of service names for which their services are either deployed or undeployed:

```
<soapenv:Body>
    <ns1:DoServiceDeployment
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
      <DeploymentServiceRequest href="#id0"/>
```

```
    </ns1:DoServiceDeployment>
    <multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns2:DeploymentServiceRequest"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
      <NodeName xsi:type="xsd:string">172.19.240.61</NodeName>
      <DeployType href="#id1"/>
      <ServiceList xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[1]">
        <item>service name</item>
      </ServiceList>
    </multiRef>
    <multiRef id="id1" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns3:DeployType"
xmlns:ns3="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">Deploy</multiRef>
  </soapenv:Body>
```

Response Format

The response contains the performed query information as defined in the previous "Response Format" section on page 2-49.

Fault

Invalid Service: Nonexisting Service

If the request to activate or deactivate a service is for an invalid service name, such as a nonexistent service name, the ReasonCode -1010 and the ReasonCodeString "No such service" will appear in the response.

The following request and response example applies for activating the service "Cisco WrongService."

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:DoServiceDeployment
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <DeploymentServiceRequest xsi:type="ns2:DeploymentServiceRequest"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <NodeName xsi:type="xsd:string">172.19.240.99</NodeName>
    <DeployType xsi:type="ns2:DeployType">Deploy</DeployType>
    <ServiceList xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[1]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
     <item>Cisco WrongService</item>
    </ServiceList>
   </DeploymentServiceRequest>
  </ns1:DoServiceDeployment>
 </soapenv:Body>
</soapenv:Envelope>


<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
```

```
  <ns1:DoServiceDeploymentResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <ServiceInformationResponse xsi:type="ns2:ServiceInformationResponse"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <ReturnCode xsi:type="ns2:ReturnCode">0</ReturnCode>
    <ReasonCode xsi:type="xsd:integer">-1010</ReasonCode>
    <ReasonString xsi:type="xsd:string">No such service</ReasonString>
    <ServiceInfoList xsi:type="ns2:ServiceInformation" xsi:nil="true"/>
   </ServiceInformationResponse>
  </ns1:DoServiceDeploymentResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Invalid Service: Nondeployable Service

If the request to activate or deactivate a service is for a nondeployable service, a service with the Deployable attribute set to False in the response for getting the static service specification, such as service “SNMP Master Agent,” the ReasonCode -1045 and the ReasonCodeString “Operation not supported” will appear in the response.

The following request and response example applies for activating the service “SNMP Master Agent.”

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:DoServiceDeployment
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <DeploymentServiceRequest xsi:type="ns2:DeploymentServiceRequest"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <NodeName xsi:type="xsd:string">172.19.240.99</NodeName>
    <DeployType xsi:type="ns2:DeployType">Deploy</DeployType>
    <ServiceList xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[1]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
     <item>SNMP Master Agent</item>
    </ServiceList>
   </DeploymentServiceRequest>
  </ns1:DoServiceDeployment>
 </soapenv:Body>
</soapenv:Envelope>

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:DoServiceDeploymentResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <ServiceInformationResponse xsi:type="ns2:ServiceInformationResponse"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <ReturnCode xsi:type="ns2:ReturnCode">0</ReturnCode>
    <ReasonCode xsi:type="xsd:integer">-1045</ReasonCode>
    <ReasonString xsi:type="xsd:string">Operation not supported</ReasonString>
    <ServiceInfoList xsi:type="ns2:ServiceInformation" xsi:nil="true"/>
   </ServiceInformationResponse>
  </ns1:DoServiceDeploymentResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Invalid Service: Empty List of Services

If the request to activate or deactivate a service provides an empty list of services, the ReasonCode -1045 and the ReasonCodeString “Operation not supported” will appear in the response.

The following request and response example applies for activating the service without providing any service name:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:DoServiceDeployment
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <DeploymentServiceRequest xsi:type="ns2:DeploymentServiceRequest"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <NodeName xsi:type="xsd:string">172.19.240.99</NodeName>
    <DeployType xsi:type="ns2:DeployType">Deploy</DeployType>
    <ServiceList xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[0]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
   </DeploymentServiceRequest>
  </ns1:DoServiceDeployment>
 </soapenv:Body>
</soapenv:Envelope>

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:DoServiceDeploymentResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <ServiceInformationResponse xsi:type="ns2:ServiceInformationResponse"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <ReturnCode xsi:type="ns2:ReturnCode">0</ReturnCode>
    <ReasonCode xsi:type="xsd:integer">-1045</ReasonCode>
    <ReasonString xsi:type="xsd:string">Operation not supported</ReasonString>
    <ServiceInfoList xsi:type="ns2:ServiceInformation" xsi:nil="true"/>
   </ServiceInformationResponse>
  </ns1:DoServiceDeploymentResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Request Example

The request example for deploying “Cisco Tftp” service follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:DoServiceDeployment
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
      <DeploymentServiceRequest href="#id0"/>
    </ns1:DoServiceDeployment>
```

```
    <multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns2:DeploymentServiceRequest"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
      <NodeName xsi:type="xsd:string">172.19.240.61</NodeName>
      <DeployType href="#id1"/>
      <ServiceList xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[1]">
        <item>Cisco Tftp</item>
      </ServiceList>
    </multiRef>
    <multiRef id="id1" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns3:DeployType"
xmlns:ns3="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">Deploy</multiRef>
  </soapenv:Body>
</soapenv:Envelope>
```

Response Example

The response example for deploying “Cisco Tftp” service follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:DoServiceDeploymentResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <ServiceInformationResponse xsi:type="ns2:ServiceInformationResponse"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <ReturnCode xsi:type="ns2:ReturnCode">0</ReturnCode>
    <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
    <ReasonString xsi:type="xsd:string" xsi:nil="true"/>
    <ServiceInfoList xsi:type="soapenc:Array"
soapenc:arrayType="ns2:ServiceInformation[1]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
     <item>
      <ServiceName xsi:type="xsd:string">Cisco Tftp</ServiceName>
      <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
      <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
      <ReasonCodeString xsi:type="xsd:string"> </ReasonCodeString>
      <StartTime xsi:type="xsd:string">Wed Sep 15 13:59:28 2004</StartTime>
      <UpTime xsi:type="xsd:integer">6</UpTime>
     </item>
    </ServiceInfoList>
   </ServiceInformationResponse>
  </ns1:DoServiceDeploymentResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Control Services API

This **soapDoControlServices** API allows clients to start or stop a list of service. It returns the services response information for the services that are requested to get started or stopped. The API does not allow clients to stop the following non-stop services:

- A Red Hat DB
- Cisco Tomcat
- Cisco Database Layer Monitor

- Cisco Unified Communications Manager Serviceability
- Cisco SOAP -Real-Time Service APIs
- Cisco SOAP -Performance Monitoring APIs
- Cisco SOAP -Log Collection APIs

The API also does not allow clients to provide an empty list of services when it tries to start or stop the services.

Request Format

HTTP header should have following SOAP action and envelop information:

```
<operation name="soapDoControlServices">
    <soap:operation
soapAction="http://schemas.cisco.com/ast/soap/action/#ControlCenterServices#soapDoCont
rolServices"/>
    <input>
        <soap:body use="encoded" namespace="http://schemas.cisco.com/ast/soap/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output>
        <soap:body use="encoded" namespace="http://schemas.cisco.com/ast/soap/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
</operation>
```

The **soapDoControlServices** operation takes one array of service names for which their services are either started or stopped.

```
<multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns2:ControlServiceRequest"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
      <NodeName xsi:type="xsd:string" xsi:nil="true"/>
      <ControlType href="#id1"/>
      <ServiceList xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[1]">
        <item>service name</item>
      </ServiceList>
    </multiRef>
    <multiRef id="id1" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns3:ControlType"
xmlns:ns3="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">Start</multiRef>
```

Response Format

The response contains the performed query information as defined in the previous "Response Format" section on page 2-49.

Fault

Invalid Service: Nonexisting Service

If the request of start/stop service for an invalid service name, such as a nonexisting service name, the ReasonCode -1010 and the ReasonCodeString “No such service” will be the response. The following request and response example applies for starting the service “Cisco WrongService.”

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:DoControlServices
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <ControlServiceRequest xsi:type="ns2:ControlServiceRequest"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <NodeName xsi:type="xsd:string" xsi:nil="true"/>
    <ControlType xsi:type="ns2:ControlType">Start</ControlType>
    <ServiceList xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[1]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
     <item>Cisco WrongService</item>
    </ServiceList>
   </ControlServiceRequest>
  </ns1:DoControlServices>
 </soapenv:Body>
</soapenv:Envelope>


<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:DoControlServicesResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <ServiceInformationResponse xsi:type="ns2:ServiceInformationResponse"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <ReturnCode xsi:type="ns2:ReturnCode">0</ReturnCode>
    <ReasonCode xsi:type="xsd:integer">-1010</ReasonCode>
    <ReasonString xsi:type="xsd:string">No such service</ReasonString>
    <ServiceInfoList xsi:type="ns2:ServiceInformation" xsi:nil="true"/>
   </ServiceInformationResponse>
  </ns1:DoControlServicesResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Invalid Service: Nonstop Service

If the request of stop service for a nonstop service, such as service “Cisco Tomcat”, the ReasonCode -1045 and the ReasonCodeString “Operation not supported” will be the response. The following request and response example applies for stopping the service “Cisco Tomcat.”

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:DoControlServices
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
```

```
   <ControlServiceRequest xsi:type="ns2:ControlServiceRequest"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <NodeName xsi:type="xsd:string" xsi:nil="true"/>
    <ControlType xsi:type="ns2:ControlType">Stop</ControlType>
    <ServiceList xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[1]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
     <item>Cisco Tomcat</item>
    </ServiceList>
   </ControlServiceRequest>
  </ns1:DoControlServices>
 </soapenv:Body>
</soapenv:Envelope>

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:DoControlServicesResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <ServiceInformationResponse xsi:type="ns2:ServiceInformationResponse"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <ReturnCode xsi:type="ns2:ReturnCode">0</ReturnCode>
    <ReasonCode xsi:type="xsd:integer">-1045</ReasonCode>
    <ReasonString xsi:type="xsd:string">Operation not supported</ReasonString>
    <ServiceInfoList xsi:type="ns2:ServiceInformation" xsi:nil="true"/>
   </ServiceInformationResponse>
  </ns1:DoControlServicesResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Invalid Service: Empty List of Services

If the request of start or stop service is with an empty list of services, the ReasonCode -1045 and the ReasonCodeString “Operation not supported” will be the response. The following request and response example applies for stopping the service without providing any service name.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:DoControlServices
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <ControlServiceRequest xsi:type="ns2:ControlServiceRequest"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <NodeName xsi:type="xsd:string" xsi:nil="true"/>
    <ControlType xsi:type="ns2:ControlType">Stop</ControlType>
    <ServiceList xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[0]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
   </ControlServiceRequest>
  </ns1:DoControlServices>
 </soapenv:Body>
</soapenv:Envelope>

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:DoControlServicesResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
```

```
   <ServiceInformationResponse xsi:type="ns2:ServiceInformationResponse"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
    <ReturnCode xsi:type="ns2:ReturnCode">0</ReturnCode>
    <ReasonCode xsi:type="xsd:integer">-1045</ReasonCode>
    <ReasonString xsi:type="xsd:string">Operation not supported</ReasonString>
    <ServiceInfoList xsi:type="ns2:ServiceInformation" xsi:nil="true"/>
   </ServiceInformationResponse>
  </ns1:DoControlServicesResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Invalid Service: Service with Stopping Status

If the request is to stop a service, and the service status is Stopping, the ReasonCode -1045 and the ReasonCodeString “Operation not supported” will be the response. The request and response example appears very similar to the preceding example.

Request Example

The request example for starting “Cisco Tftp” service is:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:DoControlServices
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
      <ControlServiceRequest href="#id0"/>
    </ns1:DoControlServices>
    <multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns2:ControlServiceRequest"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
      <NodeName xsi:type="xsd:string" xsi:nil="true"/>
      <ControlType href="#id1"/>
      <ServiceList xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[1]">
        <item>Cisco Tftp</item>
      </ServiceList>
    </multiRef>
    <multiRef id="id1" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns3:ControlType"
xmlns:ns3="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">Start</multiRef>
  </soapenv:Body>
</soapenv:Envelope>
```

Response Example

The response example for starting “Cisco Tftp” service follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
```

```
    <ns1:DoControlServicesResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
      <ServiceInformationResponse xsi:type="ns2:ServiceInformationResponse"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
        <ReturnCode xsi:type="ns2:ReturnCode">0</ReturnCode>
        <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
        <ReasonString xsi:type="xsd:string" xsi:nil="true"/>
        <ServiceInfoList xsi:type="soapenc:Array"
soapenc:arrayType="ns2:ServiceInformation[1]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
          <item>
            <ServiceName xsi:type="xsd:string">Cisco Tftp</ServiceName>
            <ServiceStatus xsi:type="ns2:ServiceStatus">Started</ServiceStatus>
            <ReasonCode xsi:type="xsd:integer">-1</ReasonCode>
            <ReasonCodeString xsi:type="xsd:string">
            </ReasonCodeString>
            <StartTime xsi:type="xsd:string">Tue Sep 14 19:36:08 2004</StartTime>
            <UpTime xsi:type="xsd:integer">6</UpTime>
          </item>
        </ServiceInfoList>
      </ServiceInformationResponse>
    </ns1:DoControlServicesResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

GetProductInformationList API

The GetProductInformationList API provides information on the products that are installed on a given server. This information includes

- Active Server Version
- Primary Node name
- SecondaryNode name (if any)
- Array Of Installed Products

  Each installed product provides the following information:

  - ProductName
  - Product Version
  - Product Description
  - Product ID
  - Short Name for the product
- Array Of Product Service Specification

  Each Product Service Specification provides the following information:

  - Service Name
  - Service Type
  - Deployable value
  - GroupName
  - ProductID
  - Array of DependentServices (if any).

For each server in the cluster, clients are expected to send one request to get all this information. The system requires clients to send this request only once during initialization.

Request Format

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:GetProductInformationList
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
      <ServiceInformationResponse
xsi:type="xsd:string">getProduct</ServiceInformationResponse>
    </ns1:GetProductInformationList>
  </soapenv:Body>
</soapenv:Envelope>
```

Response Format

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:GetProductInformationListResponse
         soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:ns1="http://schemas.cisco.com/ast/soap/">
      <GetProductInformationListResponse xsi:type="ns2:GetProductInformationListResponse"
         xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
        <ActiveServerVersion xsi:type="xsd:string">6.0.0.9381-5</ActiveServerVersion>
        <PrimaryNode xsi:type="xsd:string">irv3-ccm4</PrimaryNode>
        <SecondaryNode xsi:type="xsd:string">
        </SecondaryNode>
        <Products soapenc:arrayType="ns2:InstalledProduct[3]"
                  xsi:type="soapenc:Array"
                  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
          <item xsi:type="ns2:InstalledProduct">
            <ProductName xsi:type="xsd:string">Cisco Unified Cisco Unified Communications
                                               Manager</ProductName>
            <ProductVersion xsi:type="xsd:string">6.0.0.9381-5</ProductVersion>
            <ProductDescription xsi:type="xsd:string">Cisco Unified Communications Manager
                                           temporary description</ProductDescription>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
                                             Manager</ProductID>
            <ShortName xsi:type="xsd:string">CUCM</ShortName>
          </item>
          <item xsi:type="ns2:InstalledProduct">
            <ProductName xsi:type="xsd:string">Cisco Unity Connection</ProductName>
            <ProductVersion xsi:type="xsd:string">6.0.0.9381-5</ProductVersion>
            <ProductDescription xsi:type="xsd:string">Unity Connection temporary
                                                      description</ProductDescription>
            <ProductID xsi:type="xsd:string">UnityConnection</ProductID>
            <ShortName xsi:type="xsd:string">CUC</ShortName>
          </item>

          <item xsi:type="ns2:InstalledProduct">
            <ProductName xsi:type="xsd:string">Common Services</ProductName>
            <ProductVersion xsi:type="xsd:string">6.0.0.9381-5</ProductVersion>
            <ProductDescription xsi:type="xsd:string">Common Services for all
                                                      Products</ProductDescription>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <ShortName xsi:type="xsd:string">SYS</ShortName>
          </item>
        </Products>
```

```
        <Services soapenc:arrayType="ns2:ProductServiceSpecification[58]"
                  xsi:type="soapenc:Array"
                  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Unified Communications
                                               Manager</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
                                             Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Tftp</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
                                             Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
        ..
        ..
        </Services>
      </GetProductInformationListResponse>
    </ns1:GetProductInformationListResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Fault

The system issues a standard SOAP fault in case of failure.

Example

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:GetProductInformationList
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
      <ServiceInformationResponse xsi:type="xsd:string">test</ServiceInformationResponse>
    </ns1:GetProductInformationList>
  </soapenv:Body>
</soapenv:Envelope>
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:GetProductInformationListResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
      <GetProductInformationListResponse xsi:type="ns2:GetProductInformationListResponse"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/ControlCenterServices/">
        <ActiveServerVersion xsi:type="xsd:string">6.0.0.9381-5</ActiveServerVersion>
        <PrimaryNode xsi:type="xsd:string">irv3-ccm4</PrimaryNode>
        <SecondaryNode xsi:type="xsd:string">
```

```
        </SecondaryNode>
        <Products soapenc:arrayType="ns2:InstalledProduct[3]" xsi:type="soapenc:Array"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
          <item xsi:type="ns2:InstalledProduct">
            <ProductName xsi:type="xsd:string">Cisco Unified Cisco Unified Communications
Manager</ProductName>
            <ProductVersion xsi:type="xsd:string">6.0.0.9381-5</ProductVersion>
           <ProductDescription xsi:type="xsd:string">Cisco Unified Communications Manager
temporary description</ProductDescription>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <ShortName xsi:type="xsd:string">CCM</ShortName>
          </item>
          <item xsi:type="ns2:InstalledProduct">
            <ProductName xsi:type="xsd:string">Cisco Unity Connection</ProductName>
            <ProductVersion xsi:type="xsd:string">6.0.0.9381-5</ProductVersion>
            <ProductDescription xsi:type="xsd:string">Unity Connection temporary
description</ProductDescription>
            <ProductID xsi:type="xsd:string">UnityConnection</ProductID>
            <ShortName xsi:type="xsd:string">CUC</ShortName>
          </item>
          <item xsi:type="ns2:InstalledProduct">
            <ProductName xsi:type="xsd:string">Common Services</ProductName>
            <ProductVersion xsi:type="xsd:string">6.0.0.9381-5</ProductVersion>
            <ProductDescription xsi:type="xsd:string">Common Services for all
Products</ProductDescription>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <ShortName xsi:type="xsd:string">SYS</ShortName>
          </item>
        </Products>
        <Services soapenc:arrayType="ns2:ProductServiceSpecification[58]"
xsi:type="soapenc:Array" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Cisco Unified Communications
Manager</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Tftp</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Messaging Interface</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco IP Voice Media Streaming
App</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
```

```
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco CTIManager</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices soapenc:arrayType="xsd:string[1]" xsi:type="soapenc:Array">
              <item xsi:type="xsd:string">Cisco Cisco Unified Communications
Manager</item>
            </DependentServices>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Cisco Unified Communications Manager
Attendant Console Server</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CTI Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Extension Mobility</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco IP Manager Assistant</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CTI Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Extended Functions</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">Voice Quality Reporter Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Serviceability Reporter</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">Performance and Monitoring
Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
```

```
            <item xsi:type="ns2:ProductServiceSpecification">
              <ServiceName xsi:type="xsd:string">Cisco CTL Provider</ServiceName>
              <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
              <Deployable xsi:type="xsd:boolean">true</Deployable>
              <GroupName xsi:type="xsd:string">Security Services</GroupName>
              <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
              <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
            </item>
            <item xsi:type="ns2:ProductServiceSpecification">
              <ServiceName xsi:type="xsd:string">Cisco WebDialer Web Service</ServiceName>
              <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
              <Deployable xsi:type="xsd:boolean">true</Deployable>
              <GroupName xsi:type="xsd:string">CTI Services</GroupName>
              <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
              <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
            </item>
            <item xsi:type="ns2:ProductServiceSpecification">
              <ServiceName xsi:type="xsd:string">Cisco Certificate Authority Proxy
Function</ServiceName>
              <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
              <Deployable xsi:type="xsd:boolean">true</Deployable>
              <GroupName xsi:type="xsd:string">Security Services</GroupName>
              <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
              <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
            </item>
            <item xsi:type="ns2:ProductServiceSpecification">
              <ServiceName xsi:type="xsd:string">Cisco SOAP - CDROnDemand
Service</ServiceName>
              <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
              <Deployable xsi:type="xsd:boolean">true</Deployable>
              <GroupName xsi:type="xsd:string">CDR Services</GroupName>
              <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
              <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
            </item>
            <item xsi:type="ns2:ProductServiceSpecification">
              <ServiceName xsi:type="xsd:string">Cisco CAR Scheduler</ServiceName>
              <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
              <Deployable xsi:type="xsd:boolean">true</Deployable>
              <GroupName xsi:type="xsd:string">CDR Services</GroupName>
              <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
              <DependentServices soapenc:arrayType="xsd:string[1]" xsi:type="soapenc:Array">
                <item xsi:type="xsd:string">Cisco CAR Web Service</item>
              </DependentServices>
            </item>
            <item xsi:type="ns2:ProductServiceSpecification">
              <ServiceName xsi:type="xsd:string">Cisco CAR Web Service</ServiceName>
              <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
              <Deployable xsi:type="xsd:boolean">true</Deployable>
              <GroupName xsi:type="xsd:string">CDR Services</GroupName>
              <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
              <DependentServices soapenc:arrayType="xsd:string[1]" xsi:type="soapenc:Array">
                <item xsi:type="xsd:string">Cisco CAR Scheduler</item>
              </DependentServices>
            </item>
            <item xsi:type="ns2:ProductServiceSpecification">
              <ServiceName xsi:type="xsd:string">Cisco Cisco Unified Communications Manager
SNMP Service</ServiceName>
              <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
```

```
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">Performance and Monitoring
Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco DirSync</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">Directory Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco AXL Web Service</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">Database and Admin Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Bulk Provisioning
Service</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">Database and Admin Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Dialed Number Analyzer</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco DHCP Monitor Service</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">CM Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices soapenc:arrayType="xsd:string[1]" xsi:type="soapenc:Array">
              <item xsi:type="xsd:string">Cisco Database Layer Monitor</item>
            </DependentServices>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco TAPS Service</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">true</Deployable>
            <GroupName xsi:type="xsd:string">Database and Admin Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
```

```
    <ServiceName xsi:type="xsd:string">A Cisco DB</ServiceName>
    <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
    <Deployable xsi:type="xsd:boolean">false</Deployable>
    <GroupName xsi:type="xsd:string">Platform Services</GroupName>
    <ProductID xsi:type="xsd:string">Common</ProductID>
    <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
  </item>
  <item xsi:type="ns2:ProductServiceSpecification">
    <ServiceName xsi:type="xsd:string">A Cisco DB Replicator</ServiceName>
    <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
    <Deployable xsi:type="xsd:boolean">false</Deployable>
    <GroupName xsi:type="xsd:string">Platform Services</GroupName>
    <ProductID xsi:type="xsd:string">Common</ProductID>
    <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
  </item>
  <item xsi:type="ns2:ProductServiceSpecification">
    <ServiceName xsi:type="xsd:string">Cisco Tomcat</ServiceName>
    <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
    <Deployable xsi:type="xsd:boolean">false</Deployable>
    <GroupName xsi:type="xsd:string">Platform Services</GroupName>
    <ProductID xsi:type="xsd:string">Common</ProductID>
    <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
  </item>
  <item xsi:type="ns2:ProductServiceSpecification">
    <ServiceName xsi:type="xsd:string">SNMP Master Agent</ServiceName>
    <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
    <Deployable xsi:type="xsd:boolean">false</Deployable>
    <GroupName xsi:type="xsd:string">Platform Services</GroupName>
    <ProductID xsi:type="xsd:string">Common</ProductID>
    <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
  </item>
  <item xsi:type="ns2:ProductServiceSpecification">
    <ServiceName xsi:type="xsd:string">Cisco Database Layer Monitor</ServiceName>
    <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
    <Deployable xsi:type="xsd:boolean">false</Deployable>
    <GroupName xsi:type="xsd:string">DB Services</GroupName>
    <ProductID xsi:type="xsd:string">Common</ProductID>
    <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
  </item>
  <item xsi:type="ns2:ProductServiceSpecification">
    <ServiceName xsi:type="xsd:string">MIB2 Agent</ServiceName>
    <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
    <Deployable xsi:type="xsd:boolean">false</Deployable>
    <GroupName xsi:type="xsd:string">Platform Services</GroupName>
    <ProductID xsi:type="xsd:string">Common</ProductID>
    <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
  </item>
  <item xsi:type="ns2:ProductServiceSpecification">
    <ServiceName xsi:type="xsd:string">Host Resources Agent</ServiceName>
    <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
    <Deployable xsi:type="xsd:boolean">false</Deployable>
    <GroupName xsi:type="xsd:string">Platform Services</GroupName>
    <ProductID xsi:type="xsd:string">Common</ProductID>
    <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
  </item>
  <item xsi:type="ns2:ProductServiceSpecification">
    <ServiceName xsi:type="xsd:string">Native Agent Adapter</ServiceName>
    <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
    <Deployable xsi:type="xsd:boolean">false</Deployable>
    <GroupName xsi:type="xsd:string">Platform Services</GroupName>
    <ProductID xsi:type="xsd:string">Common</ProductID>
    <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
  </item>
  <item xsi:type="ns2:ProductServiceSpecification">
```

```
            <ServiceName xsi:type="xsd:string">System Application Agent</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">Platform Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco CDP Agent</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">Platform Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Syslog Agent</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">Platform Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Cisco Unified Communications Manager
Admin</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">System Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Unified Serviceability</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">System Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Unified RTMT Servlet</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">Performance and Monitoring</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco RTMT Reporter Servlet</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">Performance and Monitoring</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Cisco Unified Communications Manager
Personal Directory</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">System Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
```

```
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Log Partition Monitoring
Tool</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">System Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices soapenc:arrayType="xsd:string[1]" xsi:type="soapenc:Array">
              <item xsi:type="xsd:string">Cisco Database Layer Monitor</item>
            </DependentServices>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco CDP</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">System Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">SOAP -Real-Time Service APIs</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">SOAP Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">SOAP -Performance Monitoring
APIs</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">SOAP Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">SOAP -Log Collection APIs</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">SOAP Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Electronic Notification</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">Platform Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco License Manager</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">Platform Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco DRF Master</ServiceName>
```

```
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">Backup and Restore Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco DRF Local</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">Backup and Restore Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Certificate Expiry
Monitor</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">Platform Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Trace Collection
Servlet</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">System Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Trace Collection
Service</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">System Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Tomcat Stats Servlet</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">Performance and Monitoring</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco CDR Repository Manager</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">CDR Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices soapenc:arrayType="xsd:string[2]" xsi:type="soapenc:Array">
              <item xsi:type="xsd:string">Cisco Database Layer Monitor</item>
              <item xsi:type="xsd:string">A Cisco DB</item>
            </DependentServices>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco CDR Agent</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
```

```
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">CDR Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices soapenc:arrayType="xsd:string[2]" xsi:type="soapenc:Array">
              <item xsi:type="xsd:string">Cisco Database Layer Monitor</item>
              <item xsi:type="xsd:string">A Cisco DB</item>
            </DependentServices>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco RIS Data Collector</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">System Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco AMC Service</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Service</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">System Services</GroupName>
            <ProductID xsi:type="xsd:string">Common</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Extension Mobility
Application</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">System Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
          <item xsi:type="ns2:ProductServiceSpecification">
            <ServiceName xsi:type="xsd:string">Cisco Cisco Unified Communications Manager
Cisco IP Phone Services</ServiceName>
            <ServiceType xsi:type="ns2:ServiceTypes">Servlet</ServiceType>
            <Deployable xsi:type="xsd:boolean">false</Deployable>
            <GroupName xsi:type="xsd:string">System Services</GroupName>
            <ProductID xsi:type="xsd:string">Cisco Unified Communications
Manager</ProductID>
            <DependentServices xsi:type="xsd:string" xsi:nil="true"/>
          </item>
        </Services>
      </GetProductInformationListResponse>
    </ns1:GetProductInformationListResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Log Collection Service

This section describes the API calls for the log collection service.

ListNodeServiceLogs

listNodeServiceLogs returns array of NodeServiceLogList. This array includes the node names in the cluster and the lists of service names and system log names.

Request Format

The input is a dummy ListRequest Handle.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:ListNodeServiceLogs
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <ListRequest href="#id0"/>
  </ns1:ListNodeServiceLogs>
  <multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns2:ListRequest" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/LogCollection/">handle</multiRef>
 </soapenv:Body>
</soapenv:Envelope>
```

Response Format

The response is an array of NodeServiceLogList.

```
message="impl:listNodeServiceLogsResponse" name="listNodeServiceLogsResponse" />
String name;
String[] serviceLog;
String[] systemlog;

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:ListNodeServiceLogsResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <ListNodeServiceLogs xsi:type="soapenc:Array"
soapenc:arrayType="ns2:NodeServiceLogList[2]"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/LogCollection/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
    <item>
     <name xsi:type="xsd:string">172.19.240.92</name>
     <ServiceLog xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[40]">
      <item>Cisco Syslog Agent</item>
      <item>Cisco Unified CM SNMP Service</item>
      <item>Cisco CDP Agent</item>
      <item>Cisco CDP</item>
```

```
        <item>Cisco Log Partition Monitoring Tool</item>
        <item>Cisco RIS Data Collector</item>
        <item>Cisco AMC Service</item>
        <item>Cisco Serviceability Reporter</item>
        <item>Cisco Unified CM Admin Web Service</item>
        <item>Cisco Unified CM Realm Web Service</item>
        <item>Cisco Unified CM Service Web Service</item>
        <item>Cisco SOAP Web Service</item>
        <item>Cisco RTMT Web Service</item>
        <item>Cisco CAR Web Service</item>
        <item>Cisco Unified CM PD Web Service</item>
        <item>Cisco Unified CM DBL Web Library</item>
        <item>Cisco Unified CM NCS Web Library</item>
        <item>Cisco Unified Communications Manager</item>
        <item>Cisco Unified IP Phone Services</item>
        <item>Cisco AXL Web Service</item>
        <item>Cisco WebDialer Web Service</item>
        <item>Cisco WebDialerRedirector Web Service</item>
        <item>Cisco Ccmuser Web Service</item>
        <item>Cisco Extended Functions</item>
        <item>Cisco CDR Repository Manager</item>
        <item>Cisco CDR Agent</item>
        <item>Cisco CAPF</item>
        <item>Cisco CTLProvider</item>
        <item>Cisco Unified Communications Manager</item>
        <item>Cisco DirSync</item>
        <item>Cisco CTIManager</item>
        <item>Cisco TFTP</item>
        <item>Cisco Ip Voice Media Streaming App</item>
        <item>CMI</item>
        <item>Cisco Database Layer Monitor</item>
        <item>Cisco Car Scheduler</item>
        <item>Cisco Ipma Services</item>
        <item>Cisco Extension Mobility</item>
        <item>Database Layer (DBL) Logs</item>
        <item>Prog Logs</item>
        <item>SQL DBMS Logs</item>
       </ServiceLog>
       <Systemlog xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[5]">
        <item>Event Viewer-Application Log</item>
        <item>Install Logs</item>
        <item>Event Viewer-System Log</item>
        <item>Security Logs</item>
        <item>Cisco Tomcat</item>
       </Systemlog>
     </ListNodeServiceLogs>
    </ns1:ListNodeServiceLogsResponse>
   </soapenv:Body>
  </soapenv:Envelope>
```

Fault

If the database is not up and running or no node exists in the database, it will return a remote exception, "Query selectAllProcessNodes failed."

If no service list or syslog list gets returned from the preferences, the system will return a remote exception: “No service found” or ”No System Log found.”

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:ListNodeServiceLogsResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <ListNodeServiceLogs xsi:type="soapenc:Array"
soapenc:arrayType="ns2:NodeServiceLogList[2]"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/LogCollection/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
    <item>
     <name xsi:type="xsd:string">172.19.240.92</name>
     <ServiceLog xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[40]">
      <item>Cisco Syslog Agent</item>
      <item>Cisco Unified CM SNMP Service</item>
      <item>Cisco CDP Agent</item>
      <item>Cisco CDP</item>
      <item>Cisco Log Partition Monitoring Tool</item>
      <item>Cisco RIS Data Collector</item>
      <item>Cisco AMC Service</item>
      <item>Cisco Serviceability Reporter</item>
      <item>Cisco Unified CM Admin Web Service</item>
      <item>Cisco Unified CM Realm Web Service</item>
      <item>Cisco Unified CM Service Web Service</item>
      <item>Cisco SOAP Web Service</item>
      <item>Cisco RTMT Web Service</item>
      <item>Cisco CAR Web Service</item>
      <item>Cisco Unified CM PD Web Service</item>
      <item>Cisco Unified CM DBL Web Library</item>
      <item>Cisco Unified CM NCS Web Library</item>
      <item>Cisco Unified Communications Manager Cisco Unified IP Phone
Services</item>
      <item>Cisco AXL Web Service</item>
      <item>Cisco WebDialer Web Service</item>
      <item>Cisco WebDialerRedirector Web Service</item>
      <item>Cisco Ccmuser Web Service</item>
      <item>Cisco Extended Functions</item>
      <item>Cisco CDR Repository Manager</item>
      <item>Cisco CDR Agent</item>
      <item>Cisco CAPF</item>
      <item>Cisco CTLProvider</item>
      <item>Cisco Unified Communications Manager</item>
      <item>Cisco DirSync</item>
      <item>Cisco CTIManager</item>
      <item>Cisco TFTP</item>
      <item>Cisco Ip Voice Media Streaming App</item>
      <item>CMI</item>
      <item>Cisco Database Layer Monitor</item>
      <item>Cisco Car Scheduler</item>
      <item>Cisco Ipma Services</item>
      <item>Cisco Extension Mobility</item>
      <item>Database Layer (DBL) Logs</item>
      <item>Prog Logs</item>
      <item>SQL DBMS Logs</item>
     </ServiceLog>
     <Systemlog xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[5]">
      <item>Event Viewer-Application Log</item>
      <item>Install Logs</item>
      <item>Event Viewer-System Log</item>
```

```
      <item>Security Logs</item>
      <item>Cisco Tomcat</item>
     </Systemlog>
   </ListNodeServiceLogs>
  </ns1:ListNodeServiceLogsResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

SelectLogFiles

selectLogFiles takes FileSelectionCriteria object as an input parameter and returns FileSelectionResult. In FileSelectionCriteria, you must specify the files that you need to download by using these parameters:

- LogCollectionService URL - http://hostname/logcollectionservice/services/LogCollectionPort
- Array of Service Names / System Log Names for which the files need to be collected
- Username -CMAdministrator
- Password - ciscocisco
- Frequency - OnDemand
- File Collection Time range - Possible values are Day, Week, Hour, Minute, Month
- File Collection Time Value - Possible values are 1 -100.
- Time Zone - Example - "Client:(GMT-8:0)Pacific Standard Time"
- Time to wait after sending a request - in milliseconds
- Local folder where the files are to copied - Example d:\soaptest
- Hostname of the server

Request Format

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:SelectLogFiles
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <FileSelectionCriteria href="#id0"/>
  </ns1:SelectLogFiles>
  <multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns2:SchemaFileSelectionCriteria"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/LogCollection/">
   <ServiceLogs xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[45]">
    <item>Cisco Syslog Agent</item>
    <item>Cisco Unified CM SNMP Service</item>
    <item>Cisco CDP Agent</item>
    <item>Cisco CDP</item>
    <item>Cisco Log Partition Monitoring Tool</item>
    <item>Cisco RIS Data Collector</item>
    <item>Cisco AMC Service</item>
    <item>Cisco Serviceability Reporter</item>
    <item>Cisco Unified CM Admin Web Service</item>
    <item>Cisco Unified CM Realm Web Service</item>
    <item>Cisco Unified CM Service Web Service</item>
```

```
      <item>Cisco SOAP Web Service</item>
      <item>Cisco RTMT Web Service</item>
      <item>Cisco CAR Web Service</item>
      <item>Cisco Unified CM PD Web Service</item>
      <item>Cisco Unified CM DBL Web Library</item>
      <item>Cisco Unified CM NCS Web Library</item>
      <item>Cisco Unified Communications Manager Cisco Unified IP Phone Services</item>
      <item>Cisco AXL Web Service</item>
      <item>Cisco WebDialer Web Service</item>
      <item>Cisco WebDialerRedirector Web Service</item>
      <item>Cisco Ccmuser Web Service</item>
      <item>Cisco Extended Functions</item>
      <item>Cisco CDR Repository Manager</item>
      <item>Cisco CDR Agent</item>
      <item>Cisco CAPF</item>
      <item>Cisco CTLProvider</item>
      <item>Cisco Unified Communications Manager</item>
      <item>Cisco DirSync</item>
      <item>Cisco CTIManager</item>
      <item>Cisco TFTP</item>
      <item>Cisco Ip Voice Media Streaming App</item>
      <item>CMI</item>
      <item>Cisco Database Layer Monitor</item>
      <item>Cisco Car Scheduler</item>
      <item>Cisco Ipma Services</item>
      <item>Cisco Extension Mobility</item>
      <item>Database Layer (DBL) Logs</item>
      <item>Prog Logs</item>
      <item>SQL DBMS Logs</item>
      <item>Event Viewer-Application Log</item>
      <item>Install Logs</item>
      <item>Event Viewer-System Log</item>
      <item>Security Logs</item>
      <item>Cisco Tomcat</item>
     </ServiceLogs>
     <SystemLogs xsi:type="xsd:string" xsi:nil="true"/>
     <SearchStr xsi:type="xsd:string" xsi:nil="true"/>
     <Frequency href="#id1"/>
     <ToDate xsi:type="xsd:string" xsi:nil="true"/>
     <FromDate xsi:type="xsd:string" xsi:nil="true"/>
     <TimeZone xsi:type="xsd:string">Client:(GMT-8:0)Pacific Standard Time</TimeZone>
     <RelText href="#id2"/>
     <RelTime xsi:type="xsd:byte">5</RelTime>
     <Port xsi:type="xsd:byte">0</Port>
     <IPAddress xsi:type="xsd:string">MCS-SD4</IPAddress>
     <UserName xsi:type="xsd:string" xsi:nil="true"/>
     <Password xsi:type="xsd:string" xsi:nil="true"/>
     <ZipInfo xsi:type="xsd:boolean">false</ZipInfo>
    </multiRef>
    <multiRef id="id2" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns3:RelText"
xmlns:ns3="http://cisco.com/ccm/serviceability/soap/LogCollection/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">Days</multiRef>
    <multiRef id="id1" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns4:Frequency"
xmlns:ns4="http://cisco.com/ccm/serviceability/soap/LogCollection/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">OnDemand</multiRef>
   </soapenv:Body>
</soapenv:Envelope>
```

Response Format

The response returns a FileSelectionResult object, which contains the list of matching file names and their location in the server.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:SelectLogFilesResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://schemas.cisco.com/ast/soap/">
   <FileSelectionResult xsi:type="ns2:SchemaFileSelectionResult"
xmlns:ns2="http://cisco.com/ccm/serviceability/soap/LogCollection/">
    <Node xsi:type="ns2:Node">
     <name xsi:type="xsd:string">MCS-SD4</name>
     <ServiceList xsi:type="soapenc:Array" soapenc:arrayType="ns2:ServiceLogs[1]"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
      <item>
       <name xsi:type="xsd:string" xsi:nil="true"/>
       <SetOfFiles xsi:type="soapenc:Array" soapenc:arrayType="ns2:file[747]">
        <item>
         <name xsi:type="xsd:string">messages</name>
         <absolutepath
xsi:type="xsd:string">/var/log/active/syslog/messages</absolutepath>
         <filesize xsi:type="xsd:string">1048783</filesize>
        </item>
        <item>
         <name xsi:type="xsd:string">messages.1</name>
         <absolutepath
xsi:type="xsd:string">/var/log/active/syslog/messages.1</absolutepath>
         <filesize xsi:type="xsd:string">1208798</filesize>
        </item>
        <item>
         <name xsi:type="xsd:string">rtmt00025.log</name>
         <absolutepath
xsi:type="xsd:string">/var/log/active/tomcat/logs/rtmt/log4j/rtmt00025.log</absolutepath>
         <filesize xsi:type="xsd:string">1000084</filesize>
        </item>
        <item>
         <name xsi:type="xsd:string">rtmt00026.log</name>
         <absolutepath
xsi:type="xsd:string">/var/log/active/tomcat/logs/rtmt/log4j/rtmt00026.log</absolutepath>
         <filesize xsi:type="xsd:string">1000104</filesize>
        </item>
       </SetOfFiles>
      </item>
     </ServiceList>
    </Node>
   </FileSelectionResult>
   <ScheduleList xsi:type="soapenc:Array" soapenc:arrayType="ns3:Schedule[0]"
xmlns:ns3="http://cisco.com/ccm/serviceability/soap/LogCollection/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"/>
  </ns1:SelectLogFilesResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Fault

If the specified frequency is null, it will throw a remote exception, “LogCollection frequency is null.” If the array of ServiceLogs and System Logs is null, it throws a remote exception, “No Service/Syslog are provided for the collection.” If a matching file is found, it throws a remote exception, "The File Vector from the server is null”

Example

The following example shows a FileCollection from all the services in last 5 days:

```
https://172.19.240.90:8443/logcollectionservice/services/LogCollectionPort
CMAdministrator ciscocisco OnDemand Days 5 "Client:(GMT-8:0)Pacific Standard Time"
6000000 "/var/log/SoapTest" 172.19.240.90
```

For the absolute time, the request would look like this:

```
https://172.19.240.90:8443/logcollectionservice/services/LogCollectionPort
CMAdministrator ciscocisco OnDemand "11/14/04  2:15 PM" "11/15/04  2:15 PM"
"Client:(GMT-8:0)Pacific Standard Time" 6000000 "/var/log/SoapTest" 172.19.240.90
```

GetOneFile

The getOneFile API takes as an input parameter the absolute file name of the file that you want to collect from the server.

The return value specifies the file name, but the file name will have an AXIS-specific name. After the file is downloaded, you must replace it with the actual file name that you got from the server.

This APIS is in a different service, and the service name is DimeGetFileService. The URL follows: https://host:8443/logcollectionservice/services/DimeGetFileService.

Request Format

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:GetOneFile soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="DimeGetFileService">
   <FileName xsi:type="xsd:string">/var/log/active/syslog/messages</FileName>
  </ns1:GetOneFile>
 </soapenv:Body>
</soapenv:Envelope>
```

Response Format

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:GetOneFileResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="DimeGetFileService">
   <DataHandler href="cid:967B4FFE5D1E6F693815D4CA118E91D0"/>
  </ns1:GetOneFileResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Fault

None.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/env<soapenv:Body>"
    <soapenv:Body>
        <ns1:GetOneFile soapenv:encodingStyle="http://schemas.xmlsoap.org/
            <FileName xsi:type=”xsd:string”>/var/log/active/cm/trace/ris/sdi/
        </ns1:GetOneFile>
    </soapenv:Body>
</soapenv:Envelope>
```

CDR on Demand Service

The CDR On-Demand Service comprises a public SOAP/HTTPS interface that is exposed to third-party billing applications or customers to allow them to query the Cisco Unified Communications Manager CDR Repository Node to retrieve CDR/CMR files on demand through the use of two new API calls, get_file_list and get_file.

In previous releases, the CDR database stored CDR records, and third-party applications could query the database directly for the CDR records. In this release, CDRs no longer get stored in the CDR database, but as flat files.

The CDR On-Demand Service allows applications to obtain CDR files in a two-step process. First, the application requests CDR file lists based on a specific time interval; then, it can request specific CDR files from those lists that are returned via a (s)FTP session.

The billing application can acquire a list of CDR files that match a specified time interval (get_file_list), with the maximum time span being 1 hour. If the application needs to retrieve CDR files that span an interval over 1 hour, multiple get_file_list requests must be made to the servlet.

After the list of files is retrieved, the third-party application can then request a specific file (get_file). Upon receiving the request, the servlet initiates a (s)FTP session and sends the requested file to the application. Only one file per request is allowed, to avoid timeouts and other potential complications.

The CDR Repository node normally transfers CDR files to the billing servers once on a preconfigured schedule, then deletes them per the Cisco Unified Communications Manager configuration and other criteria. If for some reason the billing servers do not receive the CDR files, or want to have them sent again, they can do so using the SOAP/HTTPS CDR On-Demand APIs. After CDR files are deleted, you cannot retrieve them.

The CDR On-Demand Service provides the following features:

- API to get a list of files that match a specified time interval (get_file_list)
- API to request a specific file that matches a specified filename (get_file)
- Limit of 1300 file names get returned from the get_file_list API
- Specified time range cannot span over 1 hour
- Service not available during CDR repository file maintenance window
- CDR files are sent via standard FTP or (s)FTP, which is user configurable
- API to request specific file (get_file) can return only one file per request
- Servlet needs to be activated through Service Activation Page

Before an application can access the CDR files, you must ensure that the SOAP APIs are activated from the Service Activation window on the CDR Repository Node where the CDR Repository Manager is activated.

Step 1 Go to http://<IP Address of Unified CM node>:8080/ccmservice

Step 2 Choose **Tools →Service Activation**.

Step 3 Select the server where the CDR Repository Manager resides.

Step 4 Under the CDR Services section, start the following services:

- Cisco SOAP - CDROnDemandService
- CDR Repository Manager

The CDR On-Demand Service depends on the CDR Repository Manager, so both must be activated.

Step 5 Click **Update** and wait until the page refreshes.

Tip Remember that the On-Demand Service will not function during the Maintenance period.

Security

You can use standard FTP or SFTP to deliver the CDR files. Refer to RFC959 and RFC2228 for further details of these applications.

The CDR On-Demand Service will create either a standard FTP or SFTP session with the billing server each time that a CDR file is to be sent. Exceptions get thrown whenever an error occurs on the Servlet side. In addition, all errors will get written into log files.

On the billing application side, Cisco recommends that billing applications implement code to catch these exceptions and display the exception string for detailed error conditions.

The following sections describe the two APIs that comprise the CDR On-Demand Service.

get_file_list

The get_file_list API allows the application to query the CDR Repository Node for a list of all the files that match a specified time interval. The time interval of the request cannot exceed 1 hour. If you want a list of files that span more than the 1 hour time interval that is allowed, you must make multiple requests to the servlet to acquire multiple lists of filenames.

The get_file_list API returns an array of strings that contain the list of all the filenames that match the specified time interval. If no filenames exist that match the time range, the value that is returned from the API call is simply null. If any time errors are encountered, exceptions get thrown. In addition, logs will be kept detailing the errors. Find these log files in the /var/log/active/tomcat/logs/soap/log4j directory.

A limit of 1300 file names can be returned to the application as a result of a get_file_list API call. If the file list that is returned contains 1300 file names, but does not span the entire requested time interval, you should make additional requests with the start time of the subsequent requests as the time of the last file name that was returned in the previous request.

Parameters

The get_file_list API expects the following parameters:

Start Time

Mandatory parameter that specifies the starting time for the search interval. The format is a string: YYYYMMDDHHMM. No default value exists.

End Time

Mandatory parameter that specifies the ending time for the search interval. The format is a string: YYYYMMDDHHMM. No default value exists.

Note The time span between Start Time and End Time must constitute a valid interval, but be not longer than 1 hour.

Where to get the files from

Mandatory parameter that tells the servlet whether to include those files that were successfully sent to the third-party billing servers. The format is boolean.

- True = Include both files that were sent successfully and files that failed to be sent.
- False = Send only the files that failed to be sent. Do not include files that were sent successfully.

get_file

The get_file API allows customers to request a specific CDR file that matches the specified filename. The resulting CDR file then gets sent to the customer via standard FTP or secure FTP, depending on the third-party billing application preference. The only constraint provides that the servlet can only process one file per request.

The get_file API returns normally with no value to indicate that the file has been successfully sent to the third-party billing server. If the transfer fails for any reason, exceptions get thrown. In addition, logs detailing the errors get saved in the /var/log/active/tomcat/logs/soap/log4j directory.

Parameters

The get_file API expects the following parameters:

Host Name

Mandatory parameter (string) that specifies the hostname of the third-party billing application server, information that the servlet needs to connect to the billing server to deliver the CDR files.

User Name

Mandatory parameter (string) that specifies the username for the third-party billing application server, information that the servlet needs to connect to the billing server to deliver the CDR files.

Password

Mandatory parameter (string) that specifies the password for the third-party billing application server, information that the servlet needs to connect to the billing server to deliver the CDR files.

Remote Directory

Mandatory parameter (string) that specifies the remote directory on the third-party billing application server to that the CDR servlet is to send the CDR files.

File Name

Mandatory parameter (string) that specifies the filename of the CDR file that the third-party billing application wants delivered from the CDR On-Demand Service.

Secure FTP

Mandatory parameter (Boolean) that specifies whether to use standard FTP or secure (s)FTP to deliver the CDR files. This depends on the third-party billing application configuration and preferences.

Fault

The CDR On-Demand Service throws exceptions when certain error conditions are met:

- The servlet gets used during the maintenance period.
- The values that are entered for starting and ending time do not reflect the correct length – 12 bytes in the format YYYYMMDDHHMM is the correct length.
- The starting and ending time spans an interval of more than 1 hour.
- The starting time is greater than or equal to the ending time (invalid interval).
- No files exist in the CDR Repository.
- The (s)FTP connection to the remote node did not get established.
- The (s)FTP application failed to send the files that the third-party billing application requested.

The exception string describes the error condition that the billing application can print with the toString() function.

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="http://schemas.cisco.com/ast/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://schemas.cisco.com/ast/soap/"
xmlns:intf="http://schemas.cisco.com/ast/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <wsdl:types>
- <schema targetNamespace="http://schemas.cisco.com/ast/soap/"
xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
- <complexType name="ArrayOf_xsd_string">
- <complexContent>
- <restriction base="soapenc:Array">
  <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]" />
  </restriction>
  </complexContent>
  </complexType>
  </schema>
  </wsdl:types>
  <wsdl:message name="get_fileResponse" />
```

```
- <wsdl:message name="get_file_listResponse">
  <wsdl:part name="get_file_listReturn" type="impl:ArrayOf_xsd_string" />
  </wsdl:message>
- <wsdl:message name="get_file_listRequest">
  <wsdl:part name="in0" type="xsd:string" />
  <wsdl:part name="in1" type="xsd:string" />
  <wsdl:part name="in2" type="xsd:boolean" />
  </wsdl:message>
- <wsdl:message name="get_fileRequest">
  <wsdl:part name="in0" type="xsd:string" />
  <wsdl:part name="in1" type="xsd:string" />
  <wsdl:part name="in2" type="xsd:string" />
  <wsdl:part name="in3" type="xsd:string" />
  <wsdl:part name="in4" type="xsd:string" />
  <wsdl:part name="in5" type="xsd:boolean" />
  </wsdl:message>
- <wsdl:portType name="CDRonDemand">
- <wsdl:operation name="get_file_list" parameterOrder="in0 in1 in2">
  <wsdl:input message="impl:get_file_listRequest" name="get_file_listRequest" />
  <wsdl:output message="impl:get_file_listResponse" name="get_file_listResponse" />
  </wsdl:operation>
- <wsdl:operation name="get_file" parameterOrder="in0 in1 in2 in3 in4 in5">
  <wsdl:input message="impl:get_fileRequest" name="get_fileRequest" />
  <wsdl:output message="impl:get_fileResponse" name="get_fileResponse" />
  </wsdl:operation>
  </wsdl:portType>
- <wsdl:binding name="CDRonDemandSoapBinding" type="impl:CDRonDemand">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="get_file_list">
  <wsdlsoap:operation
soapAction="http://schemas.cisco.com/ast/soap/action/#CDRonDemand#get_filelist" />
- <wsdl:input name="get_file_listRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://schemas.cisco.com/ast/soap/" use="encoded" />
  </wsdl:input>
- <wsdl:output name="get_file_listResponse">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://schemas.cisco.com/ast/soap/" use="encoded" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="get_file">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="get_fileRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:CDRonDemand" use="encoded" />
  </wsdl:input>
- <wsdl:output name="get_fileResponse">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://schemas.cisco.com/ast/soap/" use="encoded" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="CDRonDemandService">
- <wsdl:port binding="impl:CDRonDemandSoapBinding" name="CDRonDemand">
  <wsdlsoap:address
location="http://irv3-ccm1:8080/CDRonDemandService/services/CDRonDemand" />
  </wsdl:port>
  </wsdl:service>
  </wsdl:definitions>
```

Developer Tools

Each of the following Web Services includes a separate JSP page that can be referenced during development as developer support:

- Real-time Service — https:/<server>:8443/realtimeservice
- Performance Service — https:/<server>:8443/perfmonservice
- ControlCenter Service — https:<server>:8443/controlcenterservice
- Log Collection Service — https:/<server>:8443/logcollectionservice
- CDR On-Demand Service — https:/<server>:8443/CDROnDemandService

Note You must enter the name of each Web Service exactly as shown above.

Each Web Service JSP page offers three options:

1. View Deployed Web Services
2. View <Web Service> WSDL
3. SOAP Monitor

View Deployed Web Services

This option displays a window that is similar to Figure 2-3 for each Web Service. The *(wsdl)* links display the code for the item(s) t are listed. For more information about viewing the web services, see the *Cisco Unified Communications Manager Administration Guide.*

Figure 2-3 Deployed Web Services Window

View <Web Service> WSDL

This options displays a window similar to Figure 2-4, which displays the WSDL code for the selected web service.

Figure 2-4 Web Service WSDL Window

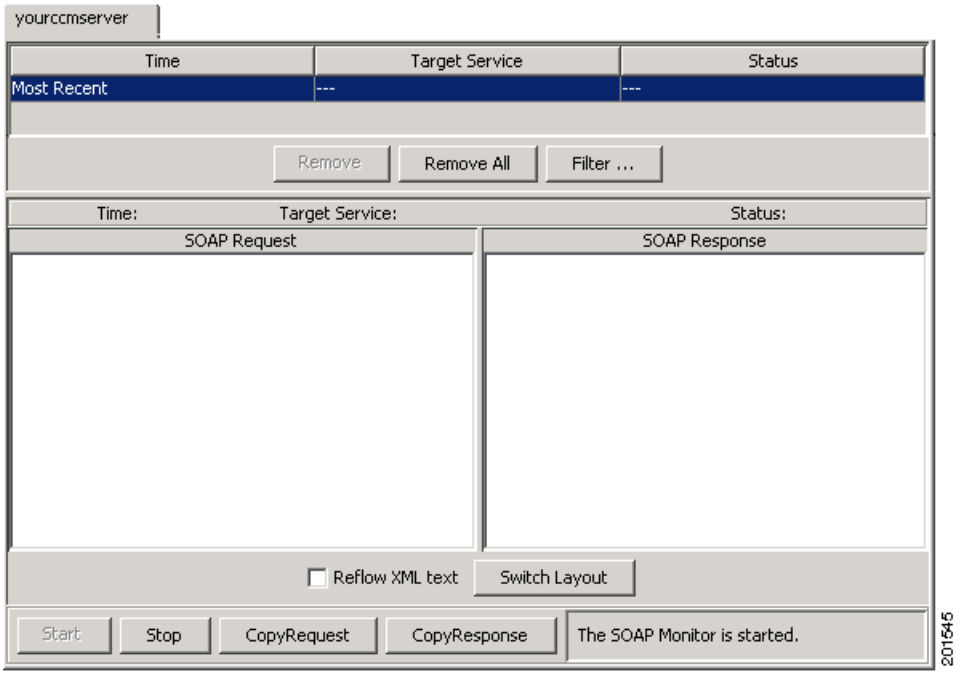
```
- <definitions name="RISService" targetNamespace="http://schemas.cisco.com/ast/soap/">
  - <!--
     ================================================
          <
          <                        XML Schemas
          <
          <================================================
    -->
  - <types>
    - <schema elementFormDefault="qualified" targetNamespace="http://schemas.cisco.com/ast/soap/">
      - <simpleType name="RisReturnCode">
        - <restriction base="string">
            <enumeration value="Ok"/>
            <enumeration value="NotFound"/>
            <enumeration value="InvalidRequest"/>
            <enumeration value="InternalError"/>
            <enumeration value="NodeNotResponding"/>
            <enumeration value="InvalidNodeName"/>
          </restriction>
        </simpleType>
      - <simpleType name="CmSelectBy">
        - <restriction base="string">
            <enumeration value="Name"/>
            <enumeration value="IpAddress"/>
            <enumeration value="DirNumber"/>
            <enumeration value="Description"/>
          </restriction>
        </simpleType>
```

201547

SOAP Monitor

This options displays the SOAP Monitor Window as illustrated in Figure 2-5. The SOAP Monitor window helps you look at the request and response messages during the development cycle.

Figure 2-5 SOAP Monitor Window



Password Expiration and Lockout

If a Credential Policy is defined for SOAP accounts to lock out with a count of login failures, the SOAP services will return “http 401 Unauthorized.”

If a Credential Policy is defined for SOAP accounts to expire, SOAP services will return “http 403 Forbidden.”

If a Credential Policy is defined for SOAP accounts to change the password, SOAP services will return “http 403 Forbidden.”

Application Customization for Cisco Unity Connection Servers

Be aware that some Serviceability SOAP operations are only available when the server runs the Cisco Unified Communications Manager software. Applications that support multiple server configurations (servers that run the Cisco Unified Communications Manager software, or the Cisco Unity Connection software, or both) must use the getProductInformation() interface to determine whether the operation they want to perform is available.

You will find that the following Serviceability SOAP operations are not available when only the Cisco Unity Connection software resides on the server:

Port or Service	Unavailable Operations
PerfmonPort	SelectCmDevice
CDROnDemand	*All operations*

The following Serviceability SOAP operations may provide different sets of target objects, depending on which software resides on the server:

Port or Service	Operations with Different Sets of Target Objects
PerfmonPort	perfmonAddCounter perfmonRemoveCounter perfmonCollectSessionData perfmonListInstance perfmonQueryCounterDescription perfmonListCounter perfmonCollectCounterData
ControlCenterServices	soapGetStaticServiceList soapGetServiceStatus soapDoServiceDeployment soapDoControlServices
LogCollectionPort	listNodeServiceLogs selectLogFiles DimeGetOneFile

CHAPTER 3

Cisco Extension Mobility Service API

The Cisco Extension Mobility service, a feature of Cisco Unified Communications Manager, allows a device, usually a Cisco Unified IP Phone, to temporarily embody a new device profile, including lines, speed dials, and services. It enables users to temporarily access their individual Cisco Unified IP Phone configuration, such as their line appearances, services, and speed dials, from other Cisco Unified IP Phones. The Cisco Extension Mobility service works by downloading a new configuration file to the phone. Cisco Unified Communications Manager dynamically generates this new configuration file based on information about the user who is logging in.

The system associates each Cisco Extension Mobility user with a device profile through configuration. When a user logs in to a phone, the phone temporarily embodies the device profile for that user. The two primary functions of the Cisco Extension Mobility feature comprise authenticating the user who is logging in and generating the right configuration file from the user information.

You can view the device profile as a template for a physical device. The device profile defines the attributes of a device, but it does not associate with a physical phone. A device profile includes information such as the phone template, user locale, services to which the device is subscribed, and so on. Because it does not associate with a physical phone, it does not have information such as MAC address, location, and region. When a phone downloads a device profile, the phone retains its physical attributes such as MAC address, device location information, device CSS, and so on.

The Cisco Unified Communications Manager support for the Cisco Extension Mobility service comprises the Cisco Extension Mobility Application (EMApp) and the Cisco Extension Mobility Service (EMService) modules. The application and service modules, along with other Cisco Unified Communications Manager infrastructure such as the Database Layer (DBL), User directory (either internal or an external LDAP directory), and TFTP server, provide the Cisco Extension Mobility feature.

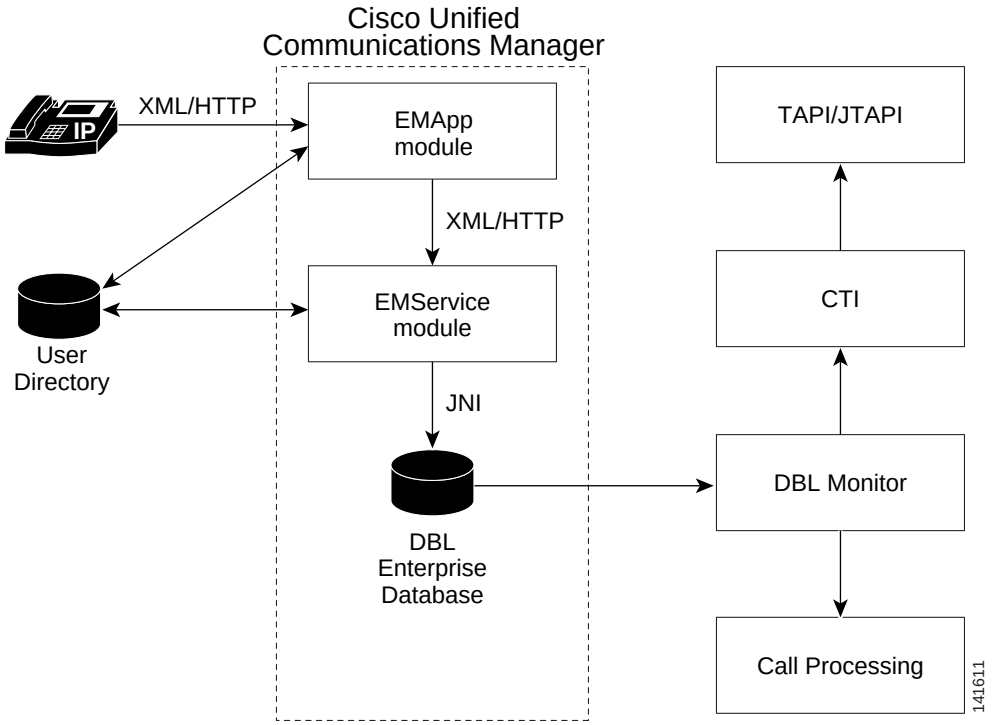
You can use the XML-based EMService API with your applications, so they can take advantage of Cisco Extension Mobility service functionality. For details about how to use the Cisco Extension Mobility service API, see the "Using the Cisco Extension Mobility API" section on page 3-6.

To successfully develop an application that uses the Cisco Extension Mobility service, you need to understand how the service operates and how your application fits into the Cisco Extension Mobility service. This chapter includes the high-level concepts that are important in understanding the Cisco Extension Mobility service:

Cisco Extension Mobility Architecture

This section explains the Cisco Extension Mobility service components and how they work with your application. Figure 3-1 provides a functional diagram of the different Cisco Extension Mobility modules that is followed by a brief description of each module.

Figure 3-1 Cisco Extension Mobility Functional Diagram



Cisco Extension Mobility Service Components

The Cisco Extension Mobility service comprises three basic architectural components. The following paragraphs provide a brief description of each component:

- **Cisco Extension Mobility Application**—A software module in Cisco Unified Communications Manager that receives XML requests (via HTTP post) for login and logout of users from the phones. It interacts with other components in the system and responds with a HTTP response message to the phone. The application module validates the “user ID” and PIN in the login request by consulting either a local user directory or an external directory like LDAP. If the “user ID” and PIN are valid, it interacts with the Cisco Extension Mobility service module on behalf of the phones. After the interaction with the service module is successful, it notifies the phone to restart with a new configuration.
- **Cisco Extension Mobility Service**—A software module in Cisco Unified Communications Manager that receives XML/HTTP requests from the application module to build a new configuration file. The application module provides information such as identity of the device and the device profile for the user who is logging in. It invokes the Database Layer (DBL) via the Java Native Interface (JNI) to write the appropriate configuration file to the TFTP server.

- **Database Layer (DBL)**—Receives a request from the Cisco Extension Mobility service module and generates a new configuration file. The device profile corresponding to the user who is logging in and the physical attributes of the phone provide the basis for the new configuration file. After the new configuration file is generated, it is pushed to the Cisco Unified Communications Manager database and a change notification gets generated to the call-processing module. This notification ultimately results in the new configuration file being pushed to the TFTP server.

The Cisco Extension Mobility service comprises your application, the EMApp module, the EMService module, the Database Layer (DBL), and the ancillary items that are listed in Table 3-1.

Table 3-1 Additional Cisco Extension Mobility Service Components

Component	Description
DBL Monitor	Database Layer Monitor service notifies other processes of changes in the Cisco Unified Communications Manager database.
LDAP Directory	Lightweight Directory Access Protocol Directory (LDAP) stores information for Cisco Unified Communications Manager.
CallProcessing	CallProcessing is a Cisco Unified Communications Manager process that maintains device connections.
CTI	Computer Telephony Interface (CTI) comprises the set of processes that expose programmable APIs for call control.
TAPI/JTAPI	TAPI and JTAPI programmatic interfaces support call control.

How Cisco Extension Mobility Works

This section describes what happens when your application sends a message to the EMService to use Cisco Extension Mobility functionality.

Figure 3-1 also illustrates how the Cisco Extension Mobility components communicate with each other. The high-level message flow between different Cisco Extension Mobility components remains the same as in previous releases.

Your login application submits an XML message to the EMService servlet by using the Hypertext Transfer Protocol (HTTP). The EMService uses the LDAP directory to check the UserID and PIN in the message from the login application. If the UserID and PIN are valid, the EMService executes the request by communicating with the database layer (DBL) through JNI. For more details about how the EMService module works, see the "Cisco Extension Mobility Service Module" section that follows.

If the DBL changes the device profile for a login or logout request, it tells the DBL Monitor, which passes this information on to the CallProcessing and CTI components. CallProcessing, in turn, tells the Cisco Unified IP Phone that it needs to restart itself to load the new device profile. For more information about device profiles, see the "Device Profiles" section on page 3-5.

The CTI layer notifies JTAPI and TAPI applications that are monitoring the device or user that the application control list has changed.

If the DBL completes a transaction successfully, it tells the EMService. The EMService then sends an XML response that the transaction was successful to your login application by using HTTP.

If the transaction is not successful, the EMService sends your login application an appropriate error message.

Cisco Extension Mobility Service Module

Your login application communicates with the Cisco Extension Mobility service through the Cisco Extension Mobility Service (EMService) component.

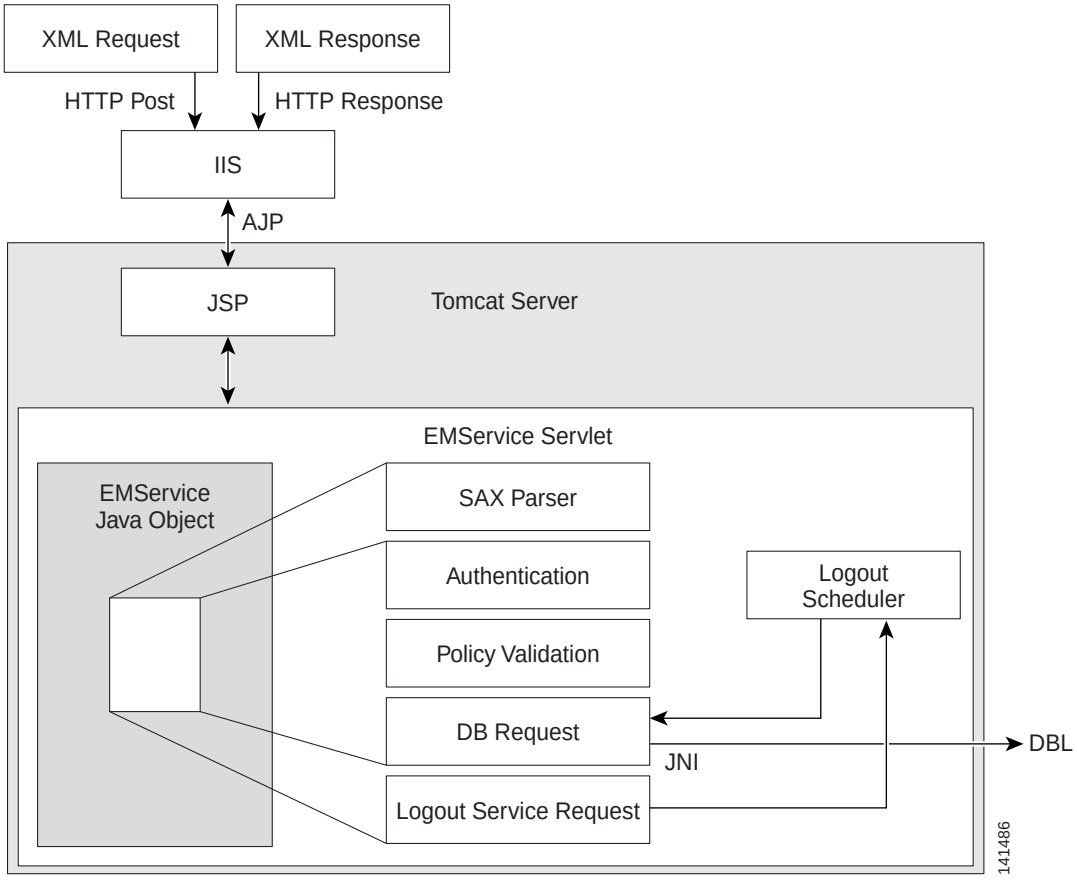
Only a single user can log in at a time on a particular device. Subsequent attempts by users to log in on a device before the previous user has logged out will fail. You also cannot log out of a device to which no user is currently logged in. These conditions generate error messages.

When the EMService component receives an HTML message from your login application, it uses HTTP to send an XML response message. The response to a request serves as success or failure message, and the response to a query serves as a query result message.

For more detailed information about these messages, see Messages, page 3-16

Figure 3-2 illustrates more details of the operation of the EMService module.

Figure 3-2 Cisco Extension Mobility EMService Module



The EMService component sends an appropriate XML error response to your login application if

- Authentication fails
- A precondition is not met
- It cannot contact the DBL
- The DBL returns an error

Authentication

The Cisco Extension Mobility service allows authentication by proxy. That is, a user with Cisco Extension Mobility proxy rights can log in any user to any device.

What this means is that an application can be responsible for authenticating a user in whatever way that the designer of the application sees fit: by using a password, PIN, hardware key, biometrics, and so on. The application must provide valid credentials for itself, so the Cisco Extension Mobility Service knows the application is provisioned in the system and allowed to log users in and out.

To this end, you must ensure that a special user that corresponds to the application is configured in the Directory. This user, representing the application, has a standard LDAP UserID and PIN. The application must send a valid UserID and PIN to log a user in or log out from a device.

Note This mechanism requires configuring a UserID and PIN for the application; you can do this by using Cisco Unified Communications Manager Administration, User Configuration.

Preconditions

The EMService Java Object Policy Validation engine checks the preconditions.

Only a single user can log in at a time on a particular device. Subsequent attempts by users to log in on a device before the previous user has logged out will fail. You also cannot log out of a device to which no user is currently logged in. These conditions generate error messages.

Automatic Logout

The Logout Scheduler in the EMService module times all login occurrences if you have specified a system maximum login time. If you have not set the login duration, the automatic logout period for that device defaults to the system maximum time.

Device Profiles

Device profiles act as the basic unit of transaction for the Cisco Extension Mobility service. A device profile contains all the configuration information, such as line appearances, speed dials, and services, for a particular device. You can think of it as a “virtual device.” It has all the properties of a device except physical characteristics such as a Media Access Control (MAC) address and a directory URL.

When a user logs in, the User device profile replaces the current device configuration. When a user logs out, the Logout device profile replaces the User device profile.

Cisco Extension Mobility requires a Logout Device Profile for each configured device. Cisco Extension Mobility uses the Logout Device Profile, which can be either an Auto-Generated or User Device Profile, as the “logged out” configuration of the device.

Two types of device profiles exist: Auto-Generated device profiles and User device profiles:

- Auto-Generated device profile—Can be used only as a Logout device profile. This provides a snapshot of the existing device configuration. You cannot associate it with a user.
- User device profile—An administrator generates it. It gets associated with a user in the same manner as any other device.

Note To create an Auto-Generated Device Profile, the system configures a device, and a snapshot of the device is taken and saved as a device profile with the prefix ADP (Auto-Generated Device Profile) and the MAC address of the device. For example, the Auto-Generated Device Profile for the device SEP000011112222 specifies ADP000011112222.

Note Cisco Extension Mobility fully supports the Cisco Unified IP Phone 7960 and the Cisco Unified IP Phone 7940 but not the Cisco IP Phone model 7910 and preceding devices.

Using the Cisco Extension Mobility API

The Cisco Extension Mobility service provides a fairly rich API, which enables extension mobility on Cisco Unified IP Phones and allows application control over authentication, scheduling, and availability.

An application that uses Cisco Extension Mobility service represents an IP phone service that allows a user to enter a userID and PIN at the phone itself and log in to the phone. The architecture and implementation of Cisco Extension Mobility make many other applications possible; some examples follow:

- An application that automatically activates phones for employees when they reserve a particular desk for a particular time (the scheduling application)
- A lobby phone that does not have a line appearance until a user logs in

The Cisco Extension Mobility API gets exposed as an Extensible Markup Language (XML) interface via HTTP. The administrator of the system designates a website as the entry point to the API, and all requests and queries are made through those URLs. This website also provides the document type definitions (DTDs) that define the XML for requests, queries, and responses. This document includes the DTDs, along with examples.

The XML input gets submitted via an HTTP POST. A field named “xml” contains the XML string that defines the request or query. The response to this HTTP POST represents a pure XML response with either a success or failure indicator for a request or the response to a query.

Note The Cisco Extension Mobility API does not use the M-POST method as defined in the HTTP Extension Framework.

This section includes the following topics:

New and Changed Information

Cisco Unified Communications Manager *New and Changed Information Guide for Release 6.0(1)* describes new features and or changes that are pertinent to the Cisco Extension Mobility Service in Cisco Unified Communications Manager Release 6.0(1).

Cisco Extension Mobility Rearchitecture

This section describes new features that allow Cisco Extension Mobility login/logout to complete successfully when connectivity to the publisher server is lost. Cisco Extension Mobility will now work when the publisher server is not available. The Cisco Extension Mobility rearchitecture has a positive impact on the number of Cisco Extension Mobility logins/logouts per hour.

Modifications to the Cisco Extension Mobility architecture mean that it can support the Call Processing User Facing feature architecture. These new features do not cause any changes in end point behavior.

Feature Description

The current Cisco Extension Mobility architecture requires a database update operation on the Device table, and, depending on Administrative provisioning, multiple database delete and database insert operations might be required for the following database tables:

- DeviceNumPlanMap
- SpeedDial
- BusyLampField
- TelecasterSubscribedService
- TelecasterSubscribedServiceParameter
- DeviceAddOnModuleMap

To support database resiliency, some fields have been moved to their own tables, so they can be updated locally and replicated in a fully meshed topology. This change will break direct SQL database access by using the AXL SOAP/XML Layer for fields in columns that moved from one table to another.

The new Cisco Extension Mobility architecture provides some performance improvement over the previous release because it requires only a single database update operation on the new ExtensionMobilityDynamic table, which increases performance by reducing database stored procedure complexity in favor of increased application complexity.

Note Applications need to take into account the information present in the ExtensionMobilityDynamic table.

Cisco Unified Communications Manager adjusts database access to account for the information now available in the new ExtensionMobilityDynamic table. The Cisco Unified Communications Manager *Data Dictionary for Release 6.0(1)* contains a comparison of the database schema between version 6.0(1) and earlier releases.

The performance enhancements in Cisco Extension Mobility resolve the following defects:

- CSCsc81681 - Cisco Extension Mobility login phones change DN is slow on performance run.
- CSCsd71925 - Cisco Extension Mobility login rate for 7835 below levels of previous releases.

ExtensionMobilityDynamic Table

The Cisco Extension Mobility rearchitecture requires the creation of a new simple table that supports simple timestamp conflict resolution and is writable on all nodes in the system. Table 3-2 illustrates the columns in this new table, ExtensionMobiltyDynamic.

Table 3-2 ExtensionMobilityDynamic Table

Column Name	Type	Nulls	Default Value	TFTP Impacted
Pkid	char(36)	No	newid()	
fkdevice	char(36)	No	None	
fkenduser	char(36)	Yes	NULL	
logintime	integer	Yes	NULL	
loginduration	integer	Yes	NULL	
fkdevice_currentloginprofile[1]	char(36)	Yes	NULL	Yes
fkenduser_lastlogin	char(36)	Yes	NULL	
fkPhoneTemplate	char(36)	Yes	NULL	Yes
fkSoftkeyTemplate	char(36)	Yes	NULL	Yes
tkUserLocale	integer	Yes	NULL	Yes
UserHoldMOHAudioSourceID	integer	Yes	NULL	
tkStatus_MLPPIndicationStatus	integer	No	0	Yes
tkPreemption	integer	No	2	Yes
ignorePI	boolean	No	False	
allowCTIControlFlag	boolean	No	False	
fkCallingSearchspace_restrict	char(36)	Yes	NULL	
fkMatrix_Presence	char(36)	Yes	NULL	
fkMLPPDomain	char(36)	Yes	NULL	Yes
ctiidBase[2]	integer	No	0	
lastNumPlanIndex[2]	integer	Yes[3]	0	Yes
misMatchedLogin)[2]	boolean	No	False	Yes
versionstamp[2]	varchar(47,0)	No	see note[4]	Yes

1. Name changed from ikdevice_currentloginprofile to fkdevice_currentloginprofile, as per the database naming conventions.
2. This is a new field.
3. Highest numplanindex for DNs that are associated with this EM login
4. 0000000000-c7a6c673-7479-46b0-839e-014d3d093963

An application can determine whether a Cisco Extension Mobility login is active on a device by testing the following conditions:

- A record for the device exists in the ExtensionMobilityDynamic table.
- The ExtensionMobilityDynamic.logintime column is non-NULL.
- The ExtensionMobilityDynamic.fkdevice_currentloginprofile is non-NULL.

An application can determine whether a Cisco Extension Mobility logout is active on a device by testing the following conditions:

- A record for the device exists in the ExtensionMobilityDynamic table.
- The ExtensionMobilityDynamic.logintime column is NULL.
- The ExtensionMobilityDynamic.fkdevice_currentloginprofile is non-NULL.

An application can determine whether neither Cisco Extension Mobility login or logout is active on a device by testing the following conditions:

- A record for the device exists in the ExtensionMobilityDynamic table.
- The ExtensionMobilityDynamic.logintime column is NULL.
- The ExtensionMobilityDynamic.fkdevice_currentloginprofile is NULL.

Call-Processing Requirements

Call Processing uses the new ExtensionMobilityDynamic table to properly configure the device on Device reset, registration, or change notification. Cisco Unified Communications Manager registers for change notification on the ExtensionMobilityDynamic table and updates any internally cached information as required.

All the call-processing requirements that are described here assume that the device is enabled for Cisco Extension Mobility; that is, the Device.allowhotelingflag = ‘t’, (TRUE).

Upon Device reset, registration, or change notification:

- If ExtensionMobilityDynamic.fkdevice_currentprofile is NULL, Cisco Unified Communications Manager does not override any device information.
- If ExtensionMobilityDynamic.fkdevice_currentprofile is non-NULL, Cisco Unified Communications Manager uses
  - ExtensionMobilityDynamic.fkenduser to override Device.fkenduser and any other device information that is derived from Device.fkenduser that is Cisco Extension Mobility specific.
  - ExtensionMobilityDynamic.fkPhoneTemplate to override Device.fkPhoneTemplate and any other device information that is derived from Device.fkPhoneTemplate.
  - ExtensionMobilityDynamic.fkSoftkeyTemplate to override Device. fkSoftkeyTemplate and any other device information that is derived from Device.fkSoftkeyTemplate.
  - ExtensionMobilityDynamic.tkUserLocale to override Device.tkUserLocale and any other device information that is derived from Device.tkUserLocale.
  - ExtensionMobilityDynamic.UserHoldMOHAudioSourceID to override Device.UserHoldMOHAudioSourceID and any other device information that is derived from Device.UserHoldMOHAudioSourceID.
  - ExtensionMobilityDynamic.tkStatus_MLPPIndicationStatus to override Device.tkStatus_MLPPIndicationStatus and any other device information that is derived from Device.tkStatus_MLPPIndicationStatus.
  - ExtensionMobilityDynamic.tkPreemption to override Device.tkPreemption and any other device information that is derived from Device.tkPreemption.
  - ExtensionMobilityDynamic.ignorePI to override Device.ignorePI and any other device information that is derived from Device.ignorePI.
  - ExtensionMobilityDynamic.allowCTIControlFlag to override Device.allowCTIControlFlag and any other device information that is derived from Device.allowCTIControlFlag.

- ExtensionMobilityDynamic.fkCallingSearchspace_restrict to override Device.fkCallingSearchspace_restrict and any other device information that is derived from Device.fkCallingSearchspace_restrict.
- DevicePrivacyDynamic.tkStatus_CallInfoPrivate joined by ExtensionMobilityDynamic.fkdevice_currentprofile to override DevicePrivacyDynamic.tkStatus_CallInfoPrivate joined by Device.pkid and any other device information, that is derived from DevicePrivacyDynamic.tkStatus_CallInfoPrivate.
- ExtensionMobilityDynamic.fkMatrix_Presence to override Device.fkMatrix_Presence and any other device information that is derived from Device.fkMatrix_Presence.
- ExtensionMobilityDynamic.fkMLPPDomain to override Device.fkMLPPDomain and any other device information that is derived from Device.fkMLPPDomain.
- ExtensionMobilityDynamic.fkdevice_currentprofile if non-NULL, instead of Device.pkid, to join with the DeviceNumPlanMap table to associate DeviceNumPlanMap information with ExtensionMobilityDynamic.fkdevice.
  - No DeviceNumPlanMap record with DeviceNumPlanMap.numPlanIndex greater than ExtensionMobilityDynamic.lastNumPlanIndex from this result set shall be accepted.
  - Cisco Unified Communications Manager uses the preceding result set and updates the DeviceNumPlanMap.ctiid values by using the following algorithm: Update each DeviceNumPlanMap.ctiid value in the result set with a new value that is derived from ((ExtensionMobilityDynamic.ctiibase * 256) + DeviceNumPlanMap.numplanindex).
  - Cisco Unified Communications Manager uses the preceding result set to join with the NumPlan table to associate NumPlan information with ExtensionMobilityDynamic.fkdevice.
  - Cisco Unified Communications Manager recalculates the DeviceNumPlanMap.busytrigger and DeviceNumPlanMap.maxnumcalls values that are returned with the preceding result set, if ExtensionMobilityDynamic.mismatchedlogin is set to ‘t’ (TRUE) by using an algorithm equivalent to the mismatched login algorithm that is currently in the DeviceLogin database stored procedure.
- ExtensionMobilityDynamic.fkdevice_currentprofile, if non-NULL, instead of Device.pkid, to join with
  - SpeedDial table to associate SpeedDial information with ExtensionMobilityDynamic.fkdevice.
  - BLFSpeedDial table to associate BLFSpeedDial information with ExtensionMobilityDynamic.fkdevice.
  - TelecasterSubscribedService table to associate TelecasterSubscribedService information with ExtensionMobilityDynamic.fkdevice.
  - TelecasterSubscribedParameter table to associate TelecasterSubscribedParameter information with ExtensionMobilityDynamic.fkdevice.
  - BLFDirectedCallPark table to associate BLFDirectedCallPark information with ExtensionMobilityDynamic.fkdevice.
  - DeviceAddOnModuleMap table to associate DeviceAddOnModuleMap information with ExtensionMobilityDynamic.fkdevice, except as described in the following item.
- ExtensionMobilityDynamic.fkdevice to join with the DeviceAddOnModuleMap table to associate DeviceAddOnModuleMap.specialloadinformation with ExtensionMobilityDynamic.fkdevice.

Be aware that this is required because the Administrator cannot provision special load information on a Device Profile.

  - DNDDynamic.DNDStatus joined by ExtensionMobilityDynamic.fkdevice_currentprofile to override DNDDynamic.DNDStatus that is joined by Device.pkid and any other device information, that is derived from DNDDynamic.DNDStatus.

- Cisco Unified Communications Manager updates DNDDynamic.DNDStatus that is joined by ExtensionMobilityDynamic.fkdevice_currentprofile, if it is non-NULL, instead of DNDDynamic.DNDStatus that is joined by Device.pkid, when necessary.
- Cisco Unified Communications Manager updates DevicePrivacyDynamic.tkStatus_CallInfoPrivate that is joined by ExtensionMobilityDynamic.fkdevice_currentprofile, if it is non-NULL, instead of DevicePrivacyDynamic.tkStatus_CallInfoPrivate that is joined by Device.pkid, when necessary.
- Cisco Unified Communications Manager validates the Ring Setting configuration in associated DeviceNumPlanMap records by checking the ProductSupportsFeature entries for the login model when ExtensionMobilityDynamic.misMatchedLogin is ‘t’ (TRUE).

  If the login model does not support Ring Setting, Cisco Unified Communications Manager overwrites the RingSetting configuration to be disabled.

CTI Requirements for Call Processing

Cisco Unified Communications Manager continues to provide the same information in CtiDeviceRegister Notify and CtiDeviceUnregisterNotify upon Device reset, registration, and Extensiion Mobility login/logout.

Database Requirements

The following database requirements result from the Cisco Extension Mobility rearchitecture:

- The ExtensionMobilityDynamic table contains the following columns: pkid, fkdevice, fkenduser, fkenduser_lastlogin, logintime, loginduration, fkdevice_defaultloginprofile, fkdevice_currentloginprofile, fkphonetemplate, fksoftkeytemplate, tkuserlocale, userholdmohaudiosourceid, tkstatus_mlppindicationstatus, tkpreemption, ignorepi, allowcticontrolflag, fkcallingsearchspace_restrict, fkmatrix_presence, fkmlppdomain, ctiidbase, lastnumplanindex, mismatchedlogin, and versionstamp.

  The system applies the same Cisco Extension Mobility business rules for the columns that moved from the Device table to the ExtensionMobilityDynamic table: Device.fkenduser_lastlogin , Device.logintime, Device.loginduration, Device.ikdevice_defaultprofile, and Device.ikdevice_currentloginprofile.

  The updated reset stored procedures account for the new information that is stored in the ExtensionMobilityDynamic table.
- The ExtensionMobilityDynamic table is the only table that is updated in response to Cisco Extension Mobility login/logouts. Device.fkenduser does not get used for Cisco Extension Mobility login/logout. No records in the Device table get updated in response to Cisco Extension Mobility login/logouts.
- Change notification gets provided for changes made to the ExtensionMobilityDynamic table, and the ExtensionMobilityDynamic.versionstamp column gets updated every time the record gets updated.

- An ExtensionMobilityDynamic record automatically gets created for each Device record when allowhotelingflag is set to a value of ‘t’ (TRUE). During automatic record creation, ExtensionMobilityDynamic.fkdevice gets set to Device.pkid, and all other columns get set to the default values that are listed in Table 3-2.

  All automatically generated Device Profiles get deleted during an L2 or W1.
- Database automatically deletes the ExtensionMobilityDynamic record for each Device record when allowhotelingflag is set to a value of ‘f’ (FALSE), where ExtensionMobilityDynamic.fkdevice equals Device.pkid.
- Database cascade deletes the ExtensionMobilityDynamic record for each Device record, where ExtensionMobilityDynamic.fkdevice equals Device.pkid.
- Database does not insert records into or delete records from the following tables in response to Cisco Extension Mobility login/logouts;
  - DeviceNumPlanMap
  - SpeedDial
  - BLFSpeedDial
  - BLFDirectedCallPark
  - TelecasterSubscribedService
  - TelecasterSubscribedServiceParameter
  - DeviceAddOnModuleMap.
- During a Cisco Extension Mobility login
  - ExtensionMobilityDynamic.mismatchedlogin gets set to a value of ‘t’ (TRUE) when the Device Profile that is selected is mismatched with the current device, and ExtensionMobilityDynamic.lastnumplanindex gets set to the greatest value of DeviceNumPlanMan.numplanindex, from the set of records where ExtensionMobilityDynamic.fkdevice_currentloginprofile equals DeviceNumPlanMap.fkdevice, limited by the mismatched login operation.
  - ExtensionMobilityDynamic.mismatchedlogin is set to a value of ‘f’ (FALSE) when the Device Profile that is selected is matched with the current device, and ExtensionMobilityDynamic.lastnumplanindex gets set to the greatest value of DeviceNumPlanMan.numplanindex, from the set of records where ExtensionMobilityDynamic.fkdevice_currentloginprofile equals DeviceNumPlanMap.fkdevice.
  - ExtensionMobilityDynamic.ctiibase gets set to a unique 32-bit number per node. The upper eight bits are set to 00000000, the next eight bits are set to 1nnnnnnn, where nnnnnnn is set to ProcessNode.nodeid for the node of the Extension Mobility Service that is servicing this Extension Mobility login, and the lower sixteen bits get set to a unique value for this node, within the range of (0 – 65535).
- During a Cisco Extension Mobility logout
  - ExtensionMobilityDynamic.mismatchedlogin gets set to a value of ‘f’ (FALSE).
  - If ExtensionMobilityDynamic.fkdevice_defaultdeviceprofile is non-NULL, ExtensionMobilityDynamic.fkdevice_currentloginprofile gets set equal to ExtensionMobilityDynamic.fkdevice_defaultdeviceprofile, and ExtensionMobilityDynamic.lastnumplanindex gets set to the greatest value of DeviceNumPlanMan.numplanindex, from the set of records where ExtensionMobilityDynamic.fkdevice_currentloginprofile equals DeviceNumPlanMap.fkdevice.

  - If ExtensionMobilityDynamic.fkdevice_defaultdeviceprofile is NULL, ExtensionMobilityDynamic.fkdevice_currentloginprofile gets set to its default value, and ExtensionMobilityDynamic.lastnumplanindex gets set to its default value.
  - If ExtensionMobilityDynamic.fkdevice_currentloginprofile is non-NULL, ExtensionMobilityDynamic.ctiibase gets set to a unique 32-bit number per node. The upper eight bits get set to 00000000, the next eight bits get set to 1nnnnnnn, where nnnnnnn is set to ProcessNode.nodeid for the node of the Cisco Extension Mobility Service that is servicing this Cisco Extension Mobility logout, and the lower sixteen bits get set to a unique value for this node, within the range of (0 – 65535).
  - If ExtensionMobilityDynamic.fkdevice_currentloginprofile is NULL, Database shall set ExtensionMobilityDynamic.ctiibase to its default value.

- The ExtensionMobilityDynamic.lastnumplanindex gets updated for each Cisco Extension Mobility login to indicate the greatest DeviceNumPlanMap.numplanindex from the DeviceNumPlanMap table that can be associated with the Device from the current Device Profile. This feature supports mismatched ExtensionMobility.login/logout; that is, when the selected Device Profile is for a device that differs in type from the current device.
- The Enterprise Parameter, “Synchronization Between Auto Device Profile and Phone Configuration,” gets removed from the database because this parameter is no longer required.

TFTP Requirements

See the “TFTP Impacted” column in Table 3-2 for the columns in the ExtensionMobilityDynamic table that apply to TFTP, if TFTP is involved with the device that is undergoing Cisco Extension Mobility login or logout. During Cisco Extension Mobility login or logout to one of these devices, TFTP uses all the requirements from the “Call-Processing Requirements” section on page 3-9 that are relevant to TFTP.

EMApp and EMService Requirements

EMApp and EMService requirements assume that no design change is required to work with the Cisco Extension Mobility rearchitecture. The only significant change requires that you query the table ExtensionMobilityDynamic instead of the Device table or EndUser table.

EMApp needs to figure out the login/logout status and also the last logged in user information. You now obtain these two pieces of information from the ExtensionMobilityDynamic table.

EMService does policy validation, gets device information, and selects all devices where the user has logged in. You now must query the ExtensionMobilityDynamic table to obtain this information. Also, EMAPI that is exposed by this component needs to update and query the ExtensionMobilityDynamic table.

Administration Requirements

The Administration Requirements assume that Cisco Extension Mobility is enabled on the current device.

After a Cisco Extension Mobility login

- Administration provides a link to the current Device Profile, ExtensionMobilityDynamic.fkdevice_currentdeviceprofile, which is associated with the current device to display the current operational state of the device and to the current EndUser Profile, ExtensionMobilityDynamic.fkenduser.

After a Cisco Extension Mobility logout

- If ExtensionMobilityDynamic.fkdevice_currentdeviceprofile is non-NULL, Administration provides a link to the current Device Profile, ExtensionMobilityDynamic.fkdevice_currentdeviceprofile, associated with the current device to display the current operational state of the device, and to the current EndUser Profile, ExtensionMobilityDynamic.fkenduser.
- If ExtensionMobilityDynamic.fkdevice_currentdeviceprofile is NULL, Administration does not provide a link to any Device Profile or to any EndUser Profile because ExtensionMobilityDynamic.fkenduser is also NULL.

Administration provides a Current Configuration button, which displays the current configuration of the device after a Cisco Extension Mobility login or logout when ExtensionMobilityDynamic.fkdevice_currentdeviceprofile is non-NULL.

Administration groups the Cisco Extension Mobility impact attributes into the Cisco Extension Mobility section on the Phone Configuration Page and, if a user is currently logged in, indicates that the configuration change for the attributes under the Cisco Extension Mobility section will not take effect until the user logs out.

Administration Use Case

For a device that has a Cisco Extension Mobility login that is currently active or a Cisco Extension Mobility Logout Profile that is not active, Administration displays "--- Use Current Device Settings ---." The settings that display on the device window still reflect the device settings and not the operational settings of the device because the database tables have not been modified.

Administration now provides a link to the current EndUser Profile and the current Device Profile.

- If the SysAdmin selects the EndUser Profile link, Administration displays the EndUser Profile.
- If the SysAdmin selects the Device Profile link, Administration displays the Device Profile.
- If the SysAdmin selects the Current Configuration button, Administration displays the current configuration.

CTI Requirements

The following paragraphs summarize the changes that have been made in CTI.

- CTIManager adjusted database access to account for information now available in the new ExtensionMobilityDynamic table.
- CTIManager registers for change notification on the ExtensionMobilityDynamic table and updates any internally cached information as required.
- If the ExtensionMobilityDynamic.fkdevice_currentprofile is NULL, Cisco Unified Communications Manager does not override any device information on Device reset, registration, or change notification.

- Cisco Unified Communications Manager uses ExtensionMobilityDynamic.fkenduser to determine the name of the currently logged-on user.
- On Device reset, registration, or change notification, if ExtensionMobilityDynamic.fkdevice_currentprofile is non-NULL
  - CTIManager uses ExtensionMobilityDynamic.fkdevice_currentprofile, instead of Device.pkid, to join with the DeviceNumPlanMap table to associate DeviceNumPlanMap information with ExtensionMobilityDynamic.fkdevice, and no DeviceNumPlanMap record with DeviceNumPlanMap.numPlanIndex greater then ExtensionMobilityDynamic.lastNumPlanIndex from this result set shall be accepted.
  - CTIManager calculates CtiID value for the line by using the formula, CtiID = ((ExtensionMobilityDynamic.ctiibase * 256) + DeviceNumPlanMap.numplanindex), to generate unique CtiID for the line.
  - CTIManager uses ExtensionMobilityDynamic.tkUserLocale to override Device.tkUserLocale and any other device information that was derived from Device.tkUserLocale.
  - CTIManager uses ExtensionMobilityDynamic.allowCTIControlFlag to override Device.allowCTIControlFlag.
  - CTIManager recalculates the DeviceNumPlanMap.busytrigger and DeviceNumPlanMap.maxnumcalls values if ExtensionMobilityDynamic.mismatchedlogin is set to ‘t’ (TRUE) by using an algorithm equivalent to the mismatched login algorithm that currently in the DeviceLogin database stored procedure.
  - CTIManager uses DNDDynamic.DNDStatus that is joined by ExtensionMobilityDynamic.fkdevice_currentprofile to override DNDDynamic.DNDStatus that is joined by Device.pkid and any other device information that is derived from DNDDynamic.DNDStatus.

CTI Use Case

From an application perspective, the functionality remains the same.

AXL SOAP Database Requirements

To allow Cisco Extension Mobility to become a CallProcessing User Facing Feature that is writable locally, required schema changes necessitate movement of columns from one table to another. Thus, this means that applications that employ direct SQL access to the database are not compatible with release 6.0(1). The Cisco Unified Communications Manager *Data Dictionary, Release 6.0(1)* documents all schema changes from releases 4.2 and 5.0. Tables and columns identified as present in release 4.2 or 5.0 and removed from release 6.0 are the columns where backward compatibility is broken.

AXL maintains backward compatibility in the APIs that do not use direct SQL access. AXL accesses the publisher database server specifically, instead of the local database.

CCMCIP

CCMCIP adjusts database access to account for information that is available in the ExtensionMobilityDynamic table for service button. When a user is logged in to the device, instead of retrieving the TelecasterSubscriberService records that are associated with the login device, CCMCIP retrieves the TelecasterSubscribedService records that are associated with the current login device profile.

Feature Interactions

Cisco Extension Mobility interacts with the Privacy and Do Not Disturb (DND) features.

Cisco Extension Mobility interacts with the Privacy feature as each Device Profile has a record associated with it in the DevicePrivacyDynamic table. When a Device Profile is in use and supports Privacy, updating the Privacy status will modify the record that is associated with the Device Profile and not the record that is associated with the Device.

Cisco Extension Mobility and the DND feature interact as each Device Profile has a record that is associated with it in the DNDDynamic table. When a Device Profile is in use and supports DND, updating the DND status will modify the record that is associated with the Device Profile and not the record that is associated with the Device.

Configuration

The Cisco Extension Mobility service application accompanies Cisco Unified Communications Manager. As such, all necessary Cisco Extension Mobility service API components get installed with the standard Cisco Unified Communications Manager installation.

To use the Cisco Extension Mobility service, create a device profile for the user who is logging in and for the target device. Use the following steps to configure Cisco Extension Mobility service:

- Activate the service.
- Create Extension Mobility IP phone service.
- Create a user device profile.
- Assign the user device profile to a user.
- Assign authentication proxy rights to a UserID.
- Assign UserID to the Standard EM Authentication Proxy Rights user group.
- Enable Cisco Extension Mobility and configure the default device profile on the target device. (You must enable Cisco Extension Mobility on a device-by-device basis.)
- Subscribe to Cisco Extension Mobility IP phone service on the target device and the device profile.
- Assign a logout device profile to a target device.
- Configure the system parameters (the system uses defaults if parameters are not manually configured).

Note Technically, no need exists to assign a profile to a user. The device profile can be specified at login.

For details on how to configure the User Device Profile, refer to the *Cisco Unified Communications Manager Administration Guide* or *Cisco Unified Communications Manager Features and Services Guide*.

Messages

You communicate between your login application and the Cisco Extension Mobility service by sending and receiving XML messages. The XML messages that you send must follow the rules that are set by the Message DTDs that are described in the "Message Document Type Definitions" section on page 3-17.

The default URL for login and logout requests and system queries is

http://<server>:8080/emservice/EMServiceServlet

The application sends authentication information, including an Application ID and an Application Certificate, at the start of message.

A password represents the only type of certificate that is currently supported. All messages must include a valid appID and appPassword, or they do not get processed. For examples of valid Cisco Extension Mobility messages, see the "Message Examples" section on page 3-19.

Login Requests

Login requests provide the cornerstone of this service, and currently they offer the most flexible and complex message type. The information that is required to process a login request must include the device that is to be logged in to and the UserID of the user who is logging in to that device. If a device profile other than the default device profile that has been associated with the user is to be used, you can specify that profile name. If the system is to automatically log the user out after a particular time, you can also specify that. To log out, you only need to provide the device name in the message.

Device-User Queries

A Device-User query represents a query wherein the login application specifies a list of one or more devices, and the system returns the userID of the user who is currently logged on to each device.

User-Devices Queries

A User-Devices query represents a query in which the login application specifies a list of one or more users, and the system returns the list of devices to which a particular user is currently logged in.

Message Document Type Definitions

A Message Document Type Definition (DTD) designates an XML list that specifies precisely which elements can appear in a request, query, or response document. It also specifies the contents and attributes of the elements.

You communicate between your login application and the Cisco Extension Mobility service by sending and receiving XML documents. These XML documents must follow the rules that the Message DTDs set. For examples of how Message DTDs are used, see the "Message Examples" section on page 3-19.

Request DTD

The Request DTD defines the login and logout messages that your application can send to the Cisco Exchange Mobility service.

```
<!-- login requests DTD -->
<!ELEMENT request (appInfo, (login | logout))>
<!ELEMENT appInfo (appID, appCertificate)>
<!ELEMENT appID           (#PCDATA)>
<!ELEMENT appCertificate (#PCDATA)>
<!ELEMENT login (deviceName, userID, deviceProfile?, exclusiveDuration?)>
<!ELEMENT logout (deviceName)>
<!ELEMENT deviceName      (#PCDATA)>
<!ELEMENT userID          (#PCDATA)>
```

```
<!ELEMENT deviceProfile  (#PCDATA)>
<!ELEMENT exclusiveDuration (time | indefinite)>
<!ELEMENT time           (#PCDATA)>
<!ELEMENT indefinite EMPTY>
```

Login or Logout Response DTD

Login or Logout Response DTD defines the messages that your application receives from the Cisco Extension Mobility service when it sends a login or logout request message.

```
<!-- login response DTD -->
<!ELEMENT response (success | failure)>
<!ELEMENT success EMPTY>
<!ELEMENT failure (error)>
<!ELEMENT error (#PCDATA)>
<!ATTLIST error
          code NMTOKEN #REQUIRED>
```

Query DTD

The Query DTD defines the Device-User and User-Devices messages that your application sends the Cisco Extension Mobility service to find out which user is logged in to a device or to which devices users are logged in.

```
<!-- login query DTD -->
<!ELEMENT query (appInfo, (deviceUserQuery | userDevicesQuery))>
<!ELEMENT appInfo (appID, appCertificate)>
<!ELEMENT appID          (#PCDATA)>
<!ELEMENT appCertificate (#PCDATA)>
<!ELEMENT deviceUserQuery (deviceName+)>
<!ELEMENT userDevicesQuery (userID+)>
<!ELEMENT deviceName     (#PCDATA)>
<!ELEMENT userID         (#PCDATA)>
```

Query Response DTD

The Query Response DTD defines the messages that your application receives from the Cisco Extension Mobility service when it sends the service a Device-User or User-Devices query.

```
<!-- login query results DTD -->
<!ELEMENT response (deviceUserResults | userDevicesResults | failure)>
<!ELEMENT deviceUserResults (device+)>
<!ELEMENT userDevicesResults (user+)>
<!ELEMENT device (userID | none | doesNotExist)>
<!ATTLIST device
          name NMTOKEN #REQUIRED>
<!ELEMENT user (deviceName+ | none | doesNotExist)>
<!ATTLIST user
          id NMTOKEN #REQUIRED>
<!ELEMENT userID (#PCDATA)>
<!ELEMENT deviceName (#PCDATA)>
<!ELEMENT none EMPTY>
<!ELEMENT doesNotExist EMPTY>
<!ELEMENT failure (errorMessage)>
<!ELEMENT errorMessage (#PCDATA)>
```

Message Examples

This section provides examples of various types of messages to aid in understanding how to use the message DTDs to communicate between your login application and the Cisco Extension Mobility service. Table 3-3 lists each example type, a description of the example, and a reference to the example.

Table 3-3 Message Examples

Message Type	Description	Reference
Login Request	The 7960LoginApp application requests that user rknotts be logged into device SEP003094C25B15.	Example 3-1
Login Request	The WebLoginApp application makes a login request that specifies the RyanTravelPhone profile and limits the login time to 60 minutes.	Example 3-2
Login Request	WebLoginApp requests that user rknotts be logged in to the specified device for an unlimited duration.	Example 3-3
Logout Request	The 7960LoginApp application requests that the current user be logged out of device SEP003094C25B15.	Example 3-4
Request Response	Response of Success to a login or logout request displays.	Example 3-5
Request Response	Failure response with error indicates an incorrect appID or password.	Example 3-6
Device-User Query	Message asks what user is logged into device SEP003094C25B15.	Example 3-7
User-Devices Query	Message asks to which devices user rknotts and fwragge are logged in.	Example 3-8
Device-User Response	Response displays that rknotts is the user who is logged in to device SEP003094C25B15.	Example 3-9
User-Devices Response	Response displays that rknotts is logged in to devices SEP003094C25B15 and SEP003094C25B49, and fwragge is logged in to device SEP003094C25B46.	Example 3-10

Request Examples

Request examples demonstrate three different login requests and one logout request. The login requests show a simple login message and two messages that specify options like using a particular device profile or setting a login duration.

Example 3-1 Simple Login Request

```
<request>
  <appInfo>
    <appID>7960LoginApp</appID>
    <appCertificate>password</appCertificate>
  </appInfo>
  <login>
    <deviceName>SEP003094C25B15</deviceName>
    <userID>rknotts</userID>
  </login>
</request>
```

Example 3-2 Login Request That Specifies a Profile and a Time Restriction

```
<request>
  <appInfo>
    <appID>WebLoginApp</appID>
    <appCertificate>password</appCertificate>
  </appInfo>
  <login>
    <deviceName>SEP003094C25B15</deviceName>
    <userID>rknotts</userID>
    <deviceProfile>RyanTravelPhone</deviceProfile>
    <exclusiveDuration>
      <time>60</time>
    </exclusiveDuration>
  </login>
</request>
```

Example 3-3 Login Request That Specifies an Unlimited Duration

```
<request>
  <appInfo>
    <appID>WebLoginApp</appID>
    <appCertificate>password</appCertificate>
  </appInfo>
  <login>
    <deviceName>SEP003094C25B15</deviceName>
    <userID>rknotts</userID>
    <exclusiveDuration>
      <indefinite/>
    </exclusiveDuration>
  </login>
</request>
```

Example 3-4 Logout Request

```
<request>
  <appInfo>
    <appID>7960LoginApp</appID>
    <appCertificate>password</appCertificate>
  </appInfo>
  <logout>
    <deviceName>SEP003094C25B15</deviceName>
  </logout>
</request>
```

Request Response Examples

The request response examples show a success message (for either login or logout) and a failure message that indicates the type of error that the login or logout attempt generated.

Example 3-5 Success Response

```
<response>
  <success/>
</response>
```

Example 3-6 Failure Response

```
<response>
  <failure>
    <error code="3">Could not authenticate 'appid'</error>
  </failure>
</response>
```

Query Examples

Query examples show a typical Device-User Query message and a typical User-Devices Query message that an application sent to the Cisco Extension Mobility service.

Example 3-7 Device-User Query

```
<query>
  <appInfo>
    <appID>applicationName</appID>
    <appCertificate>password</appCertificate>
  </appInfo>
  <deviceUserQuery>
    <deviceName>SEP003094C25B15</deviceName>
  </deviceUserQuery>
</query>
```

Example 3-8 User-Devices Query

```
<query>
  <appInfo>
    <appID>applicationName</appID>
    <appCertificate>password</appCertificate>
  </appInfo>
  <userDevicesQuery>
    <userID>rknotts</userID>
    <userID>fwragge</userID>
  </userDevicesQuery>
</query>
```

Query Response Examples

Query Response examples show messages that the Cisco Extension Mobility service sent to the login application after the login application has sent a Device-User query message or a User-Devices query message.

Example 3-9 Device-User Response

```
<results>
  <deviceUserResults>
    <device name="SEP003094C25B15">
      <userID>rknotts</userID>
    </device>
  <deviceUserResults>
</results>
<results>
```

Example 3-10 User-Devices Response

```
  <userDevicesResults>
    <user id="rknotts">
      <deviceName>SEP003094C25B15</deviceName>
      <deviceName>SEP003094C25B49</deviceName>
    </user>
    <user id="fwragge">
      <deviceName>SEP003094C249A6</deviceName>
    </user>
  </userDeviceResults>
</results>
```

Application and Service Error Codes

Table 3-4 shows the new error codes the Cisco Extension Mobility Application module returns as well as the error code numbers for the previous release, and describes what each error code means.

Table 3-4 Cisco Extension Mobility Application Error Codes

New Error Code	Previous Error Code	Message—Description
200	0	NO_ERROR—Successful.
201	1	**Authentication error** AUTHENTICATION_ERROR—Authentication failed for user. Shows the user login page with error title.
202	2	**Blank UserID or PIN** NULL_PARAM—Shows the user login page with error title.
203	3	**Application UserID/Pwd null** APP_AUTHENTICATION_ERROR—Unable to find registry entry for AppUserID/Password.
204	6	**Directory server error** DIR_SERV_ERROR—Exception occurred while performing diraccess.authenticateuserwithpin().
205	9	**User Profile absent or null** DEV_PROF_ERROR—User profile absent or null.
206	12	**No Device Profile associated with User** DEV_PROF_UNAVAILABLE—No Device profile associated with user.
207	100	**Device name empty or absent** NULL_DEV_ERROR—Device name empty or absent.
208	101	**Cannot connect to EM Service** LOGIN_SERVICE_CONN_ERROR—Unable to connect with EMService.

Table 3-5 shows the current (new) error codes that the Cisco Extension Mobility Service module returns as well as the error code numbers for the previous release, and describes what each error code means. Shaded rows in this table represent error codes that are no longer used.

Table 3-5 Cisco Extension Mobility Service Error Codes

New Error Code	Previous Error Code	Message—Description
0	0	**Unknown error** UNKNOWN_ERROR—Error due to some exception, largely unknown; scope for new error code when request type could not be determined.
1	1	**Error occurred on parsing request/query** VALIDATION_ERROR—XML error occurred while parsing. Because the request or query was incorrectly formed, system cannot properly process it.

Table 3-5 Cisco Extension Mobility Service Error Codes (continued)

New Error Code	Previous Error Code	Message—Description
2	2	**Error occurred on authenticating app user** AUTHENTICATION_ERROR—Could not execute user authentication. Error occurred during the user authentication process, and the validity of the appID and appPassword that were submitted cannot be confirmed.
3	3	**Invalid App User credentials** INVALID_AUTHENTICATION—The appID and/or appPassword that were provided are incorrect.
4	4	**Policy validation error** POLICY_VALIDATION_ERROR—Exception occurred while doing policy validation. Some generic issue exists regarding the determination of whether the request is allowed.
5	5	**Device does not allow logon** REQUEST_DENIED—The request has been denied (failed Policy Validation) for one or more reasons.
6	6	**Database error occurred** DB_ERROR—The Cisco Extension Mobility service received an error while trying to communicate with the Cisco Unified Communications Manager database.
7	7	LOGOUT_REQUEST_ERROR—Not used.
8	8	**Cannot determine query type** QUERY_TYPE_UNKNOWN—Could not determine query type (Device-User or User-Devices) for response generation.
9	9	**Directory User information error** DIRUSER_ERROR—Directory User Information related error; could not integrate locales/ info / device profiles.
10	10	**User does not have app proxy rights** PROXY_AUTHENT_NOT_ALLOWED—User does not have proxy authentication rights. The appID that is specified does not have rights to log in or log out other users.
11	11	**Device does not exist** DEVICE_DOES_NOT_EXIST—Device ID is not present in the database. The specified device for login or logout does not exist in the system.
12	12	**Device Profile does not exist** DEVICE_PROFILE_ERROR—Either a profile was specified that does not exist or no logout device profile was configured for the specified user.
13	13	PIPE_CREATE_FILE_ERROR—Unused
14	14	PIPE_ERROR—Unused
15	15	PIPE_BUSY—Unused
16	16	SET_PIPE_STATE_ERROR—Unused
17	17	PIPE_WRITE_ERROR—Unused

Table 3-5 Cisco Extension Mobility Service Error Codes (continued)

New Error Code	Previous Error Code	Message—Description
18	18	**Another user logged in** DEVICE_ALREADY_LOGGED_IN—The system could not log in to the specified device because another user is already logged in.
19	19	**No user logged in** DEVICE_NOT_LOGGED_IN—The system could not log out of the specified device because no user is logged in.
20	20	**Hoteling flag error** GET_DEVICE_HOTELING_FLAG_ERROR—The system could not determine whether the device allows Cisco Extension Mobility (also called "hoteling").
21	21	**Hoteling status error** GET_DEVICE_HOTELING_STATUS_ERROR—The system could not determine the current user status.
22	22	**Device does not allow logon** DEVICE_DOES_NOT_ALLOW_HOTELING—The device that is specified is not configured for Cisco Extension Mobility ("hoteling").
23	23	**User does not exist** USER_DOES_NOT_EXIST—The userID that was entered for login does not exist in the system.
24	24	**System not enabled for login** SYSTEM_NOT_ENABLED—System login not allowed.
25	25	**User logged in elsewhere** USER_LOGGED_IN_ELSEWHERE—The specified user is already logged in to a different device or is attempting to log in to a different device.
26	26	**Busy, please try again** LOGIN_THROTTLED—User login not allowed. The server is busy.

CHAPTER 4

Cisco Web Dialer API Programming

This chapter describes the Simple Object Access Protocol (SOAP) and HTML over secure HTTP (HTTPS) interfaces that are used to develop customized directory search applications for Cisco Web Dialer Version 1.2 and contains the following sections:

Definitions

Cisco Web Dialer Server	A server that hosts the Cisco Web Dialer application
Cisco Web Dialer Application	The software that is installed on a Cisco Unified Communications Manager server. It enables the click-to-dial functionality by creating hyperlinked telephone numbers in a company directory. This functionality allows users to make calls from the web page by clicking the telephone number of the person that they are trying to call. The Cisco Web Dialer application comprises two Java servlets, the Cisco Web Dialer servlet and the Redirector servlet.
Cisco Web Dialer Servlet	A Java servlet that allows Cisco Unified Communications Manager users in a specific cluster to make and end calls, as well as to access their phone and line configuration.
Redirector Servlet	A Java servlet that finds the Cisco Unified Communications Manager cluster for a request that a Cisco Web Dialer user makes. It redirects that request to the specific Cisco Web Dialer server that is located in that user Cisco Unified Communications Manager cluster.

Overview

Cisco Web Dialer, which is installed on a Cisco Unified Communications Manager server and used in conjunction with Cisco Unified Communications Manager, allows Cisco Unified IP Phone users to make calls from web and desktop applications. For example, Cisco Web Dialer uses hyperlinked telephone numbers in a company directory to allow users to make calls from a web page by clicking the telephone number of the person that they are trying to call.

The two main components of Cisco Web Dialer comprise the Cisco Web Dialer servlet and the Redirector servlet.

Cisco Web Dialer Servlet

The Cisco Web Dialer servlet, a Java servlet, allows Cisco Unified Communications Manager users in a specific cluster to make and end calls, as well as to access their phone and line configuration.

Cisco Web Dialer applications interact with the Cisco Web Dialer servlet through two interfaces:

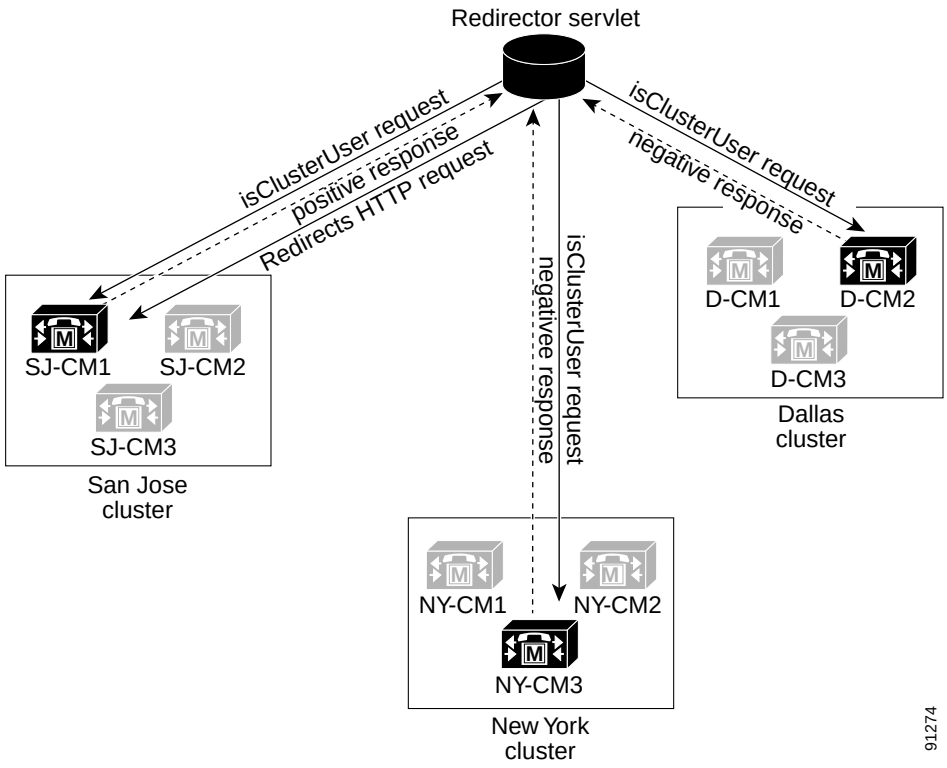
- SOAP over HTTPS —This interface, based on the Simple Object Access Protocol (SOAP), is used to develop desktop applications such as Microsoft Outlook Add-in and SameTime Client Plug-in. Developers can use the isClusterUserSoap interface to design multicluster applications that require functionality similar to a Redirector servlet.
- HTML over HTTPS—This interface, based on the HTTPS protocol, is used to develop web-based applications such as the Cisco Unified Communications Manager directory search page (directory.asp). Developers who use this interface can use the Redirector servlet for designing multicluster applications.

Redirector Servlet

The Redirector servlet, a Java-based servlet, finds the Cisco Unified Communications Manager cluster for a request that a Cisco Web Dialer user makes. It redirects that request to the specific Cisco Web Dialer server that is located in that user Cisco Unified Communications Manager cluster. Availability of the Redirector servlet occurs only for multicluster applications and only for applications that are developed by using HTML over HTTPS interfaces.

Figure 4-1 illustrates how a Redirector servlet redirects a call in a multicluster environment.

Figure 4-1 Multiple Clusters



Example of Cisco Web Dialer Using the Redirector Servlet

For example, consider three clusters, each one in a single city such as San Jose, Dallas, and New York. Each cluster contains three Cisco Unified Communications Manager servers with Cisco Web Dialer servlets that have been configured for Cisco Unified Communications Manager servers SJ-CM1, D-CM2, and NY-CM3.

The system administrator configures the Cisco Web Dialer servlets on any Cisco Unified Communications Manager server by entering the IP address of that specific Cisco Unified Communications Manager server in the *wdservers* service parameter.

For information on configuring Cisco Web Dialer and Redirector servlets, refer to the "Cisco Web Dialer" chapter in the *Cisco Unified Communications Manager Features and Services Guide, Release* 5.0.

When a user who is located in San Jose clicks a telephone number in the corporate directory search page that is enabled by Cisco Web Dialer, the following actions happen:

1. The Cisco Unified Communications Manager server sends an initial *makeCall* HTTPS request to the Redirector servlet.
2. If this request is received for the first time, the Redirector servlet reads the Cisco Web Dialer server cookie and finds it empty.

   For a repeat request, the Redirector servlet reads the IP address of the Cisco Web Dialer server that previously serviced the client and sends a *isClusterUser* HTTPS request only to that server.
3. The Redirector servlet sends back a response that asks for information, which results in the authentication dialog box opening for the user.
4. The user enters the Cisco Unified Communications Manager user ID and password and clicks the **Submit** button.

5. The Redirector servlet reads only the user identification from this information and sends a *isClusterUser* HTTPS request to each Cisco Web Dialer server that the system administrator configured.

   Figure 4-1 illustrates how this request is sent to the Cisco Web Dialer servlets that have been configured for SJ-CM1, D-CM2, and NY-CM3. Depending on the geographical location of the calling party, the Cisco Web Dialer servlet from that cluster responds positively to the Redirector servlet. The remaining Cisco Web Dialer servlets that were contacted return a negative response. The Cisco Web Dialer servlet SJ-CM1 responds positively to the request because the calling party is located in San Jose (SJ-CM).

The Redirector servlet redirects the original request from the user to SJ-CM1 and sets a cookie on the user browser for future use.

Changes in Release 5.1

Cisco Unified Communications Manager Release 5.1 includes the following change to Cisco Web Dialer:

- Cisco Web Dialer and Redirector now require HTTPS.

  Developers should format Redirector and Cisco Web Dialer requests to use HTTPS. Cisco Unified Communications Manager requires the secured protocol to prevent unauthorized applications from reading user data.

Refer to the *Cisco Unified CallManager Developers Guide for Release 5.0* for important changes to Cisco Web Dialer API programming in the 5.0 release.

Cisco Web Dialer Security Support

Cisco Web Dialer supports secure connections to CTI (TLS connection). For this feature, Cisco Web Dialer uses the security API that JTAPI provides. Refer to the *Cisco Unified Communications Manager JTAPI Developers Guide* for the JTAPI API. Cisco Web Dialer uses the Application User, "WDSysUser", for obtaining the CTI connection.

You must complete the following configuration before Cisco Web Dialer can be configured to open a CTI connection in secure mode.

Step 1 Activate the Cisco CTL Provider service in Cisco Unified Communications Manager Service Administration.

Step 2 Activate the Cisco Certificate Authority Proxy Function Service.

Step 3 Download the Cisco CTL Client from the Application plug-in and install it on any machine.

Step 4 Run the CTL Client, choose the option to "enable Cluster Security," and follow the instructions that display. This requires USB E-tokens.

Step 5 To verify that cluster security is enabled, go to Cisco Unified Communications Manager Administration and look at [System-> Enterprise Parameter configuration]. Look at the Security Parameters; the cluster security should be set to 1.

Step 6 In Cisco Unified Communications Manager Administration, from the User Management drop-down menu, select the Application User CAPF Profile option.

Step 7 Click **Add new InstanceID**.

Step 8 In the CAPF Profile configuration window, set up an InstanceID and CAPF profile for the InstanceID for the Application User WDSysUser.

a. **InstanceID**: Enter the value of instance ID; for example, 001.

b. **Certificate Operation**: Select Install/Upgrade from the drop-down menu.

c. **Authentication Mode**: Select By Authorization String from the drop-down menu.

d. **Authorization String**: Enter the value of authorization string; for example, 12345.

e. **Key Size**: Select key size from drop-down menu; for example, 1024.

f. **Operation Completes By**: Enter the date and time in following format yyyy:mm:dd:hh:mn where yyyy=year, mm=month, dd=date, hh=hour, mn=minutes, such as 2006:07:30:12:30.

Note If this date and time has passed, the certificate update operation will fail.

g. Ignore the **Packet Capture Mode**, **Packet Capture Duration**, and **Certificate** fields.

h. **Certificate Status**: Select Operation pending from the drop-down menu.

If anything else is selected, the certificate update will fail.

Security Service Parameters

Cisco Web Dialer includes two mode-specific service parameters for CTI connection security.

- **CTI Manager Connection Security Flag**—This required service parameter indicates whether security for the Cisco Web Dialer service CTI Manager connection is enabled or disabled.

  If enabled (true), Cisco Web Dialer will open a secure connection to CTI Manager by using the Application CAPF profile that is configured for the instance ID (as configured in CTI Manager Connection Instance ID service parameter) for Application user WDSysUser. The default value specifies false.

- **Application CAPF Profile Instance ID**: This service parameter specifies the Instance ID of the Application CAPF Profile for Application User WDSysUser that this Cisco Web Dialer server will use to open a secure connection to CTI Manager. You must configure this parameter if the CTI Manager Connection Security Flag parameter is enabled (true).

- **Algorithm**:

  1. Read the service parameters.
  2. Get the node IP/name of the nodes where TFTP and CAPF are activated.
  3. For the instanceID (input in service parameters), if the Certificate Operation is ‘Install/Upgrade’ or ‘Delete’, delete the current certificates, if any.
  4. If the Certificate Operation is not ‘Install/Upgrade’ or ‘Delete’, and a current certificate exists, use this certificate.
  5. If no certificate is present, request one by using JTAPI API setSecurityPropertyForInstance; this will need username, instanceID, authCode, tftpServerName, tftpPort, capfServerName, capfPort, certPath, and securityFlag. This call will contact the TFTP server, download the certificate, contact the CAPF server, verify the CTL file, and request the client and server certificates.

6. If Step 5 is successful, set the following items on the ICCNProvider and call open().provider.setInstanceID(instanceID);provider.setTFTPServer(tftpServerName);provider.setCAPFServer(capfServerName);provider.setCertificatePath(certPath);provider.setSecurityOptions(securityFlag);
7. If Step 5 fails, throw initFailedException. You can see this in the Cisco Web Dialer traces.

Install Changes

Cisco Web Dialer rpm creates a new directory "/usr/local/cm/wd/wd-certificates/" and sets permissions for users on this directory, which is used to store the certificates.

```
Files Changed: /vob/ccm/Projects/CMAppServices/rpm/cm-webdialer.spec
```

SIP Phone Support in Cisco Web Dialer

Cisco Web Dialer supports SIP phones in this release. CTI only supports the new SIP phones and not the existing SIP phones, so the same support is extended by Cisco Web Dialer. JTAPI provides the APIs that are used to distinguish between these two kinds of phones and to hide the unsupported phones from the user in the Cisco Web Dialer preferences window.

Call Flows

The call flows in this section describe the flow of events for client and browser-based applications that use Cisco Web Dialer, which should help you design customized applications for Cisco Web Dialer.

Desktop-based Client Application Call Flow

Figure 4-2 shows the call flow for an outgoing call from a client application such as Microsoft Outlook Plug-in to a Cisco Web Dialer servlet. The user clicks the **Dial** or **Make Call** button in the address book of the client application. If the user is making a call for the first time, the application does not have authentication or configuration information on the user.

If the user makes a call for the first time,

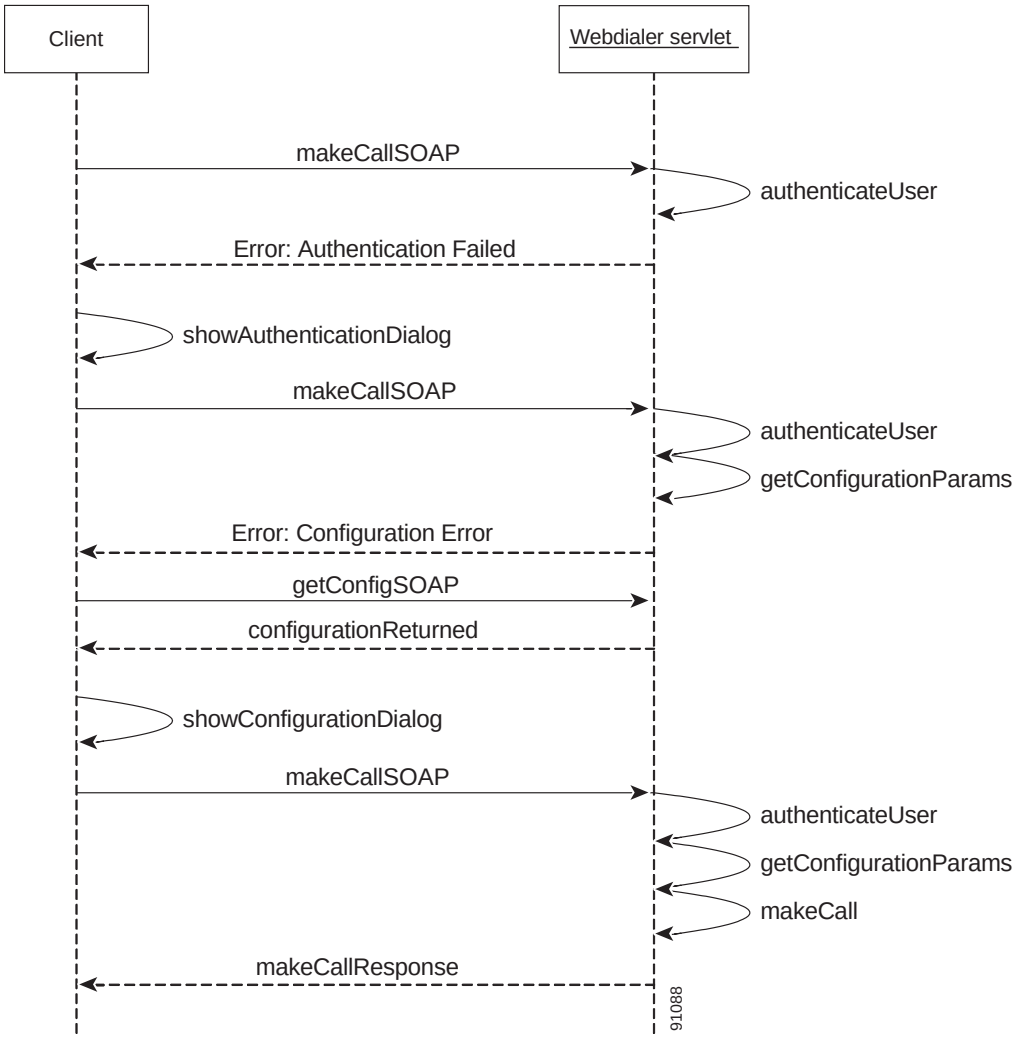
1. The client sends a makeCallSoap request to the configured Cisco Web Dialer servlet.
2. The Cisco Web Dialer servlet attempts to authenticate the user. Figure 4-2 shows an authentication failure that occurred because the authentication information is incomplete or does not exist.
3. The Cisco Web Dialer servlet sends an authentication failure response to the client application.
4. The client application displays a dialog box that asks for the user ID and password. The user enters this information and clicks the **submit** button. The user ID and password get stored for future invocations of the application.
5. The application sends a repeat SOAP request to the Cisco Web Dialer servlet. The request contains credential information on the user.
6. The Cisco Web Dialer servlet authenticates the user.
7. The Cisco Web Dialer servlet reads any missing configuration information in the request.
8. The Cisco Web Dialer servlet returns a configuration error message to the client application.
9. The client application sends a getConfigSoap request to the Cisco Web Dialer servlet.
10. The Cisco Web Dialer servlet responds with the user configuration information that is stored in the directory.

11. The client application displays a configuration dialog box that asks the user to select or update the configuration. The user enters the information and clicks the **submit** button. The user configuration information gets stored for future invocations of the application.
12. The client resends the makeCallSoap request to the Cisco Web Dialer servlet. This request contains the user configuration information.
13. The Cisco Web Dialer servlet authenticates the user and dials the telephone number by using the information that the makeCallSoap request contains. It responds to the client with a success or failure message.

Note The call flow goes directly to step 12:

- If the credential and configuration information is already stored when the application is installed.
- For all subsequent requests that the user makes.

Figure 4-2 Cisco Web Dialer Call Flow for a Client-Based Application

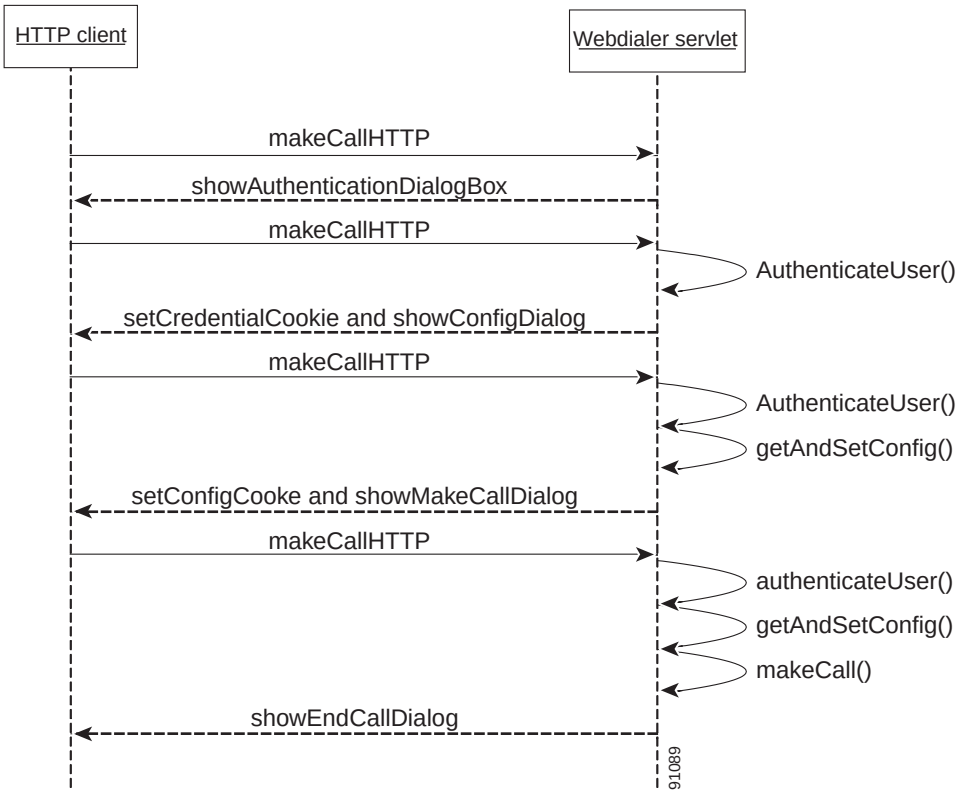


Browser-Based Application Call Flow

Figure 4-3 shows the call flow for an HTTP-based browser application such as a directory search page, personal address book, or the Cisco Unified Communications Manager directory search page (directory.asp).

The user clicks the **Dial** or **Make Call** button in the address book of the client application. If the user is making a call for the first time, the application does not have authentication or configuration information on the user.

Figure 4-3 Cisco Web Dialer Call Flow for a Browser-Based Application



If the user makes a call for the first time

1. The client sends a makeCall HTTPS request to the configured Cisco Web Dialer servlet. The query string contains the number to be called.
2. The Cisco Web Dialer servlet authenticates the user. Authentication fails because the authentication information is incomplete or does not exist.

Note Authentication succeeds if the user credentials are sent with the request, and the call flow goes directly to step 7.

3. The Cisco Web Dialer servlet sends an authentication dialog to the client browser for user authentication.
4. The user enters the user ID and password and clicks the **Submit** button.
5. The client sends a makeCallHTTPS request that contains the user credentials to the Cisco Web Dialer servlet.

6. The Cisco Web Dialer servlet authenticates the user.
7. The Cisco Web Dialer servlet reads the configuration information in the cookie that is sent with the request.
8. Assuming that the request is made for the first time, the servlet sends a response that contains a cookie to the client browser. The cookie that contains the client credentials gets stored on the client browser. The client credentials comprise user ID, IP address, and the time of the request.
9. The user enters the updates in the configuration dialog box and clicks the **Submit** button.
10. The client browser sends a makeCall HTTPS request to the Cisco Web Dialer servlet. The request contains a cookie with the credential and configuration information in parameter form.
11. The Cisco Web Dialer servlet uses the credentials to authenticate the user and saves the configuration information in its memory.
12. The Cisco Web Dialer servlet sends a makeCall confirmation dialog to the client browser with the configuration information that is stored in a cookie. The cookie gets stored on the client browser for future invocations.
13. The Make Call dialog box appears on the user computer screen. The user clicks the **Dial** button, which sends another makeCall HTTPS request to the Cisco Web Dialer servlet.
14. The Cisco Web Dialer servlet authenticates the user by using the credentials in the cookie, retrieves the configuration information from the cookie, and makes the call.
15. The servlet responds by sending an endCall confirmation dialog to the user to end the call. The End Call dialog box appears on the user computer screen and stays there for the time interval that is configured in the service parameters.

For all subsequent requests, the call flow starts at step 12 and ends at step 15.

Interfaces

Cisco Web Dialer applications interact with the Cisco Web Dialer servlet through two interfaces:

- **SOAP over HTTPS** — This interface, based on the Simple Object Access Protocol (SOAP), is used to develop desktop applications such as Microsoft Outlook Add-in and SameTime Client Plug-in. Developers can use the isClusterUserSoap interface to design multicluster applications that require functionality similar to a Redirector servlet.
- **HTML over HTTPS** — This interface, based on the HTTPS protocol, is used to develop web-based applications such as the Cisco Unified Communications Manager directory search page (directory.asp). Developers who are using this interface can use the Redirector servlet for designing multicluster applications.

Note Ensure the following files are run to properly set the ENV variable and Java classes:
installWDService.bat
installWDSOAP.bat

SOAP Over HTTPS Interface

To access the SOAP interfaces for Cisco Web Dialer, use the Cisco Web Dialer Web Service Definition Language (WSDL) in the "Cisco Web Dialer WSDL" section on page 4-17.

makeCallSoap

You access the makeCallSoap interface by initiating a SOAP request to the URL https://CCM-IP:8080/webdialer/services/WebdialerSoapService where CCM-IP specifies the IP address of the Cisco Unified Communications Manager server where Cisco Web Dialer is configured.

Parameter	Mandatory	Description	Data Type	Range Values	Default Value
Destination	Mandatory	Standard canonical form. For example, +1 408 5551212 or extensions such as 2222. The optional service parameter "Apply Application Dial Rules on SOAP Dial Request" is False by default; if this parameter is True, the destination gets transformed according to the dial rules.	String	None	None
Credential	Mandatory	The user ID or password of the user or proxy user. For more information on creating a proxy user, see the *Cisco Web Dialer* chapter in the *Cisco Unified Communications Manager Features and Services Guide, Release 5.0*.	Refer to the credential data type in the "Cisco Web Dialer WSDL" section on page 4-17.	None	None
Profile	Mandatory	The profile that is used to make a call. A typical profile is a calling device such as an IP phone or line.	Refer to the profile data type in the "Cisco Web Dialer WSDL" section on page 4-17.	None	None

Results

Refer to the "Cisco Web Dialer WSDL" section on page 4-17 for return values and their data type.

Error Code	Name	Type	Description	Action by application
0	responseCode	Integer	Success	Displays a dialog box.
	responseDescription	String	Success	
1	responseCode	Integer	Call failure error	Displays a relevant error message.
	responseDescription	String	Call failure error	

Error Code	Name	Type	Description	Action by application
2	responseCode	Integer	Authentication error	Displays the authentication dialog where the user enters ID and password information.
	responseDescription	String	User authentication error	
3	responseCode	Integer	No authentication proxy rights	Void for user-based applications.
	responseDescription	String	No authentication proxy rights	
4	responseCode	Integer	Directory error	Displays an appropriate directory error message.
	responseDescription	String	Directory error	
5	responseCode	Integer	No device is configured for the user, or missing parameters exist in the request.	The application initiates a getConfigSOAP request and displays the selected device and line to the user.
	responseDescription	String	No device is configured for the user, or missing parameters exist in the request.	
6	responseCode	Integer	Service temporarily unavailable	Displays the appropriate error dialog with an option to try again.
	responseDescription	String	Service temporarily unavailable	
7	responseCode	Integer	Destination cannot be reached.	Displays the appropriate error dialog that allows the user to edit the dialed number.
	responseDescription	String	Destination cannot be reached.	
8	responseCode	Integer	Service error	Displays the appropriate error dialog.
	responseDescription	String	Service error	
9	responseCode	Integer	Service overloaded	Displays the appropriate error dialog with an option to try again.
	responseDescription	String	Service overloaded	

This example shows a makeCallSoap request:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="urn:WebdialerSoap"
xmlns:types="urn:WebdialerSoap/encodedTypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<tns:makeCallSoap>
<cred href="#id1" />
```

```
<dest xsi:type="xsd:string">7475</dest>
<prof href="#id2" />
</tns:makeCallSoap>
<tns:Credential id="id1" xsi:type="tns:Credential">
<userID xsi:type="xsd:string">xzibit</userID>
<password xsi:type="xsd:string">55555</password>
</tns:Credential>
<tns:UserProfile id="id2" xsi:type="tns:UserProfile">
<user xsi:type="xsd:string">xzibit</user>
<deviceName xsi:type="xsd:string">SEP00131A6EC8BD</deviceName>
<lineNumber xsi:type="xsd:string">2345</lineNumber>
<supportEM xsi:type="xsd:boolean">false</supportEM>
<locale xsi:type="xsd:string" />
</tns:UserProfile>
</soap:Body>
</soap:Envelope>
```

endCallSoap

You access the endCallSoap interface by initiating a SOAP request to the URL https://CCM-IP:8080/webdialer/services/WebdialerSoapService where CCM_IP specifies the IP address of the Cisco Unified Communications Manager server where Cisco WebDialer is configured.

Parameter	Mandatory	Description	Data Type	Range Values	Default Value
Credential	Mandatory	The user ID or password of the user or proxy user. For information on creating a proxy user, see the *Cisco Web Dialer* chapter in the *Cisco Unified Communications Manager Features and Services Guide, Release 5.0*.	Refer to the credential data type in "Cisco Web Dialer WSDL" section on page 4-17.	None	None
Profile	Mandatory	The profile that is used to make a call. A typical profile is a calling device such as an IP phone or line.	Refer to the profile data type in the "Cisco Web Dialer WSDL" section on page 4-17.	None	None

Refer to the "Cisco Web Dialer WSDL" section on page 4-17 for return values and their data type.

Error Code	Name	Type	Description	Action by application
0	responseCode	Integer	Success	Displays a dialog box on the computer screen.
	responseDescription	String	Success	
1	responseCode	Integer	Call failure error	Displays a relevant error message.
	responseDescription	String	Call failure error	
2	responseCode	Integer	Authentication error	Displays authentication dialog for user to enter user ID and password.
	responseDescription	String	User authentication error	
3	responseCode	Integer	No authentication proxy rights	Void for user-based applications.
	responseDescription	String	No authentication proxy rights	
4	responseCode	Integer	Directory error	Displays an appropriate directory error message.
	responseDescription	String	Directory error	
5	responseCode	Integer	No device is configured for the user, or missing parameters exist in the request.	The Application initiates a getConfigSOAP request and displays the selected device and line to the user.
	responseDescription	String	No device is configured for the user, or missing parameters exist in the request.	
6	responseCode	Integer	Service temporarily unavailable	Displays the appropriate error dialog with an option to try again.
	responseDescription	String	Service temporarily unavailable	
7	responseCode	Integer	Destination cannot be reached.	Displays the appropriate error dialog that allows the user to edit the dialed number.
	responseDescription	String	Destination cannot be reached.	
8	responseCode	Integer	Service error	Displays appropriate error dialog.
	responseDescription	String	Service error	
9	responseCode	Integer	Service overloaded	Displays the appropriate error dialog with an option to try again.
	responseDescription	String	Service overloaded	

getProfileSoap

You access the getProfileSoap interface, which is used by plug-in based clients, by initiating a SOAP request to the URL https://CCM-IP:8080/webdialer/services/WebdialerSoapService where CCM-IP specifies the IP address of the Cisco Unified Communications Manager server where Cisco Web Dialer is configured.

Parameter	Mandatory/ Optional	Description	Data Type	Value Range	Default Value
Credential	Mandatory	User ID or password of the user or proxy user. For information on creating a proxy user, see the *Cisco Web Dialer* chapter in *Cisco Unified Communications Manager Features and Services Guide, Release 5.0*.	Refer to the credential data type in the "Cisco Web Dialer WSDL" section on page 4-17.	None	None
UserID	Mandatory	The user ID for which the configuration is requested.	String	None	None

Refer to the "Cisco Web Dialer WSDL" section on page 4-17 for return values and their data type.

Error Code	Name	Type	Description	Action by plug-in application
0	responseCode	Integer	Returns an array of phones or lines on the phone that is associated with the user. Refer to the Cisco Web Dialer WSDL for the WDDeviceInfo data type.	Displays a dialog box on the computer screen.
	responseDescription	String	Success	
	deviceInfoList	Array	Returns an array of the the WDDeviceInfo data type	
1	responseCode	Integer	No device configured for the user	Displays an appropriate error message.
	responseDescription	String	No device configured for the user	
2	responseCode	Integer	Authentication error	Displays the authentication dialog where the user enters ID and password information.
	responseDescription	String	User authentication error	
3	responseCode	Integer	No authentication proxy rights	Void for user-based applications.
	responseDescription	String	No authentication proxy rights	

Error Code	Name	Type	Description	Action by plug-in application
4	responseCode	Integer	Directory error	Displays an appropriate directory error message.
	responseDescription	String	Directory error	
6	responseCode	Integer	Service temporarily unavailable	Displays the appropriate error dialog with an option to try again.
	responseDescription	String	Service temporarily unavailable	
9	responseCode	Integer	Service overloaded	Displays the appropriate error dialog with an option to try again.
	responseDescription	String	Service overloaded	

This example shows a getProfileSoap request used for debugging purposes (normally, the SOAP implementation layer would make this request):

```
<?xml version="1.0" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="urn:WebdialerSoap"
xmlns:types="urn:WebdialerSoap/encodedTypes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <tns:getProfileSoap>
            <cred href="#id1" />
            <userid xsi:type="xsd:string">xzibit</userid>
        </tns:getProfileSoap>
        <tns:Credential id="id1" xsi:type="tns:Credential">
            <userID xsi:type="xsd:string">xzibit</userID>
            <password xsi:type="xsd:string">55555</password>
        </tns:Credential>
    </soap:Body>
</soap:Envelope>
```

isClusterUserSoap

You access the isClusterUserSoap interface by initiating a SOAP request to the URL https://CCM-IP:8080/webdialer/services/WebdialerSoapService where CCM-IP specifies the IP address of the Cisco Unified Communications Manager server where Cisco Web Dialer is configured.

Use this SOAP interface for multicluster applications that require functionality, similar to a Redirector servlet, for redirecting calls to the various locations where Cisco Web Dialer is installed on a network. The application uses this interface to locate and verify the Cisco Web Dialer that is servicing the user, followed by makeCall, endCall, or getProfile requests to that Cisco Web Dialer.

Parameter	Mandatory	Description	Data Type	Range of Values	Default Value
UserID	Mandatory	The user ID for which the the request is made.	String	None	None

Refer to the "Cisco Web Dialer WSDL" section on page 4-17 for return values and their data type.

Name	Type	Description
result	Boolean	The result specifies True if the user is present in the directory of the cluster. The result specifies False if the user is not present in the directory.

HTML Over HTTPS Interfaces

This section describes the HTML over HTTPS interfaces.

makeCall

You use the makeCall interface in customized directory search applications. The Cisco Unified Communications Manager directory search page (directory.asp) also uses this interface. Access the makeCall interface by initiating an HTTPS request to the URL https://<ipaddress>/webdialer/Webdialer. In this URL, ipaddress specifies the IP address of the Cisco Unified Communications Manager server where Cisco Web Dialer is configured.

Browser-based applications in which the browser accepts cookies use this interface. The user profile exists only for the length of the session if the cookies are disabled in a browser. For a sample script that is used to enable directory search pages, go to the "Sample JavaScript" section on page 4-20.

Parameter	Mandatory	Description	Data Type	Range of Values	Default Value
destination	Mandatory	Destination number called by the application. Number gets converted to a regular telephone number by applying the application dial rules. Refer to the *Cisco Web Dialer* chapter in the *Cisco Unified Communications Manager Features and Services Guide, Release 5.0*.	String	None	None

Name	Description
result	Cisco Web Dialer displays the appropriate dialog and its applicable success or error message. It displays an authentication dialog if no active session exists.

makeCallProxy

You access the makeCallProxy interface by initiating an HTTPS request to the URL https://ipaddress/webdialer/Webdialer?cmd=doMakeCallProxy. Browser-based applications in which the browser accepts cookies use this interface. If the cookies are disabled in a browser, the user profile exists only for the length of the session.

Applications such as a personal address book, defined in the Unified CMUser pages at https://cmserver/CMUser, can use the makeCallProxy interface. The credential of the application is used as a proxy to make calls on behalf of users. Because these users have authenticated themselves before accessing the Unified CMUser window, they do not get prompted again for their user ID and password. The application sends the user ID and password of the proxy user in the form of a query string in the request or as a parameter in the body of the POST message.

Note The API does not use the M-POST method as defined in the HTTP Extension Framework.

For a sample script that is used to enable directory search pages, go to the "Sample JavaScript" section on page 4-20.

Parameter	Mandatory	Description	Data Type	Range of Values	Default Value
uid	Mandatory	The user ID for which the request is made	String	None	None
appid	Mandatory	The userid of the application that is making a request on behalf of the user. For example, consider a Unified CM personal address book where the application allows authentication proxy rights. The appid parameter is used when the user logs in once; for example, in the Unified CM User windows. After this login, other pages do not require the user to log in again. For web page applications that are not integrated, the appid matches the userid.	String	None	None
pwd	Mandatory	The password of the appid	String	None	None
destination	Mandatory	The number to be called. The dial plan service converts this number to an E.164 number.	String	None	None

Name	Description
result	Cisco Web Dialer displays the appropriate dialog and its applicable success or error message.

Cisco Web Dialer WSDL

The WSDL specification provides the basis for the Web Service Definition Language (WSDL) for Cisco Web Dialer. You can access the WSDL for Cisco Web Dialer on the Cisco Web Dialer server installation at

https://CCM-IP:8443/webdialer/services/WebdialerSoapService?wsdl

Use this specific WSDL and the interfaces that are mentioned in this document to develop customized applications for Cisco Web Dialer. For a list of references on Cisco Unified Communications Manager, SOAP, and WSDL, refer to the "Related Documentation" section in the Preface to the *Cisco Unified CallManager Developers Guide for Release 5.0*.

```
<wsdl:definitions xmlns:tns="urn:WebdialerSoap"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="urn:WebdialerSoap" name="urn:WebdialerSoap">
    <wsdl:types>
        <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tns="urn:WebdialerSoap" targetNamespace="urn:WebdialerSoap">
            <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
            <xsd:complexType name="CallResponse">
                <xsd:sequence>
                    <xsd:element name="responseCode" type="xsd:int"/>
                    <xsd:element name="responseDescription" nillable="true"
type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="Credential">
                <xsd:sequence>
                    <xsd:element name="userID" nillable="true" type="xsd:string"/>
                    <xsd:element name="password" nillable="true" type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="UserProfile">
                <xsd:sequence>
                    <xsd:element name="user" nillable="true" type="xsd:string"/>
                    <xsd:element name="deviceName" nillable="true" type="xsd:string"/>
                    <xsd:element name="lineNumber" nillable="true" type="xsd:string"/>
                    <xsd:element name="supportEM" type="xsd:boolean"/>
                    <xsd:element name="locale" nillable="true" type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="GetConfigResponse">
                <xsd:sequence>
                    <xsd:element name="description" nillable="true" type="xsd:string"/>
                    <xsd:element name="deviceInfoList" nillable="true"
type="tns:ArrayOfWDDeviceInfo"/>
                    <xsd:element name="responseCode" type="xsd:int"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="WDDeviceInfo">
                <xsd:sequence>
                    <xsd:element name="deviceName" nillable="true" type="xsd:string"/>
                    <xsd:element name="lines" nillable="true" type="tns:ArrayOfstring"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="ArrayOfWDDeviceInfo">
                <xsd:complexContent>
                    <xsd:restriction base="soapenc:Array">
                        <xsd:attribute ref="soapenc:arrayType"
wsdl:arrayType="tns:WDDeviceInfo[]"/>
                    </xsd:restriction>
                </xsd:complexContent>
            </xsd:complexType>
            <xsd:complexType name="ArrayOfstring">
```

```
                <xsd:complexContent>
                    <xsd:restriction base="soapenc:Array">
                        <xsd:attribute ref="soapenc:arrayType"
wsdl:arrayType="xsd:string[]"/>
                    </xsd:restriction>
                </xsd:complexContent>
            </xsd:complexType>
        </xsd:schema>
    </wsdl:types>
    <wsdl:message name="makeCallSoap0In">
        <wsdl:part name="cred" type="tns:Credential"/>
        <wsdl:part name="dest" type="xsd:string"/>
        <wsdl:part name="prof" type="tns:UserProfile"/>
    </wsdl:message>
    <wsdl:message name="makeCallSoap0Out">
        <wsdl:part name="return" type="tns:CallResponse"/>
    </wsdl:message>
    <wsdl:message name="endCallSoap1In">
        <wsdl:part name="cred" type="tns:Credential"/>
        <wsdl:part name="prof" type="tns:UserProfile"/>
    </wsdl:message>
    <wsdl:message name="endCallSoap1Out">
        <wsdl:part name="return" type="tns:CallResponse"/>
    </wsdl:message>
    <wsdl:message name="getProfileSoap2In">
        <wsdl:part name="cred" type="tns:Credential"/>
        <wsdl:part name="userid" type="xsd:string"/>
    </wsdl:message>
    <wsdl:message name="getProfileSoap2Out">
        <wsdl:part name="return" type="tns:GetConfigResponse"/>
    </wsdl:message>
    <wsdl:message name="isClusterUser3In">
        <wsdl:part name="userid" type="xsd:string"/>
    </wsdl:message>
    <wsdl:message name="isClusterUser2Out">
        <wsdl:part name="return" type="xsd:boolean"/>
    </wsdl:message>
    <portType name="WebdialerSoapService">
        <wsdl:operation name="makeCallSoap">
            <wsdl:input message="tns:makeCallSoap0In"/>
            <wsdl:output message="tns:makeCallSoap0Out"/>
        </wsdl:operation>
        <wsdl:operation name="endCallSoap">
            <wsdl:input message="tns:endCallSoap1In"/>
            <wsdl:output message="tns:endCallSoap1Out"/>
        </wsdl:operation>
        <wsdl:operation name="getProfileSoap">
            <wsdl:input message="tns:getProfileSoap2In"/>
            <wsdl:output message="tns:getProfileSoap2Out"/>
        </wsdl:operation>
        <wsdl:operation name="isClusterUserSoap">
            <wsdl:input message="tns:isClusterUser3In"/>
            <wsdl:output message="tns:isClusterUser2Out"/>
        </wsdl:operation>
    </portType>
    <binding name="WebdialerSoapService" type="tns:WebdialerSoapService">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="makeCallSoap">
            <soap:operation soapAction="urn:makeCallSoap"/>
            <input>
                <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="urn:WebdialerSoap"/>
            </input>
            <output>
```

```
                <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="urn:WebdialerSoap"/>
            </output>
        </wsdl:operation>
        <wsdl:operation name="endCallSoap">
            <soap:operation soapAction="urn:endCallSoap"/>
            <input>
                <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="urn:WebdialerSoap"/>
            </input>
            <output>
                <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="urn:WebdialerSoap"/>
            </output>
        </wsdl:operation>
        <wsdl:operation name="getProfileSoap">
            <soap:operation soapAction="urn:getProfileSoap"/>
            <input>
                <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="urn:WebdialerSoap"/>
            </input>
            <output>
                <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="urn:WebdialerSoap"/>
            </output>
        </wsdl:operation>
        <wsdl:operation name="isClusterUserSoap">
            <soap:operation soapAction="urn:isClusterUserSoap"/>
            <input>
                <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="urn:WebdialerSoap"/>
            </input>
            <output>
                <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="urn:WebdialerSoap"/>
            </output>
        </wsdl:operation>
    </binding>
    <service name="WebdialerSoap">
        <port name="WebdialerSoapService" binding="tns:WebdialerSoapService">
            <soap:address
location="https://your_webdialer_server/webdialer/services/WebdialerSoapService"/>
        </port>
    </service>
</wsdl:definitions>
```

Sample JavaScript

This sample JavaScript script enables Cisco Web Dialer from a directory search page.

Single-Cluster Applications

Use this script for single-cluster applications if all users are in only one cluster.

```
function launchWebDialerWindow( url ) {
    webdialer=window.open( url, "webdialer", "status=no, width=420, height=300,
scrollbars=no, resizable=yes, toolbar=no" );
  }

  function launchWebDialerServlet( destination ) {
```

```
    url = 'https://<%=server_name%>/webdialer/Webdialer?destination=' +
escape(destination);
    launchWebDialerWindow( url );
  }
!These functions can be called from the HTML page which has a hyperlink to the phone
number to be called. An example of it is:
<TD><A href="javascript:launchWebDialerServlet( <%= userInfo.TelephoneNumber %> )"><%=
userInfo.TelephoneNumber %></A> </TD>
```

Multiple-Cluster Applications

Use this script if all users are spread across different clusters.

```
function launchWebDialerWindow( url ) {
    webdialer=window.open( url, "webdialer", "status=no, width=420, height=300,
scrollbars=no, resizable=yes, toolbar=no" );
  }

  function launchWebDialerServlet( destination ) {
    url= 'https://<%=server_name%>/webdialer/Redirector?destination='+escape(destination);
    launchWebDialerWindow( url );
  }
!These functions can be called from the HTML page which has a hyperlink to the phone
number to be called. An example of it is
<TD><A href="javascript:launchWebDialerServlet( <%= userInfo.TelephoneNumber %> )"><%=
userInfo.TelephoneNumber %></A> </TD>
```

Appendix A. AXL Configuration API List

Provided by AXLAPI.wsdl. For support, send email to developer-support@cisco.com

UCR -- Under Consideration or Review
✅ Supported, ❌ Not Supported, ℹ️ Modified

See corresponding Cisco Unified Communications Manager Developers Guide for more information.

Sorted by operation while ignoring prefixes of add, remove, update, get, and list.

Operation	3.3	4.0	4.1	4.2	5.0	5.1	5.1 (2)	6.0	UCR	Description
addAARGroup	❌	❌	❌	✅	❌	❌	❌	✅	✅	
removeAARGroup	❌	❌	❌	✅	❌	❌	❌	✅	✅	
updateAARGroup	❌	❌	❌	✅	❌	❌	❌	✅	✅	
getAARGroup	❌	❌	❌	✅	❌	❌	❌	✅	✅	
updateAARGroupMatrix	❌	❌	❌	✅	❌	❌	❌	✅	✅	
listAARGroupByName	✅	✅	✅	✅	✅	✅	✅	✅	✅	
addApplicationToSoftkeyTemplate	❌	❌	❌	✅	❌	❌	❌	✅	✅	
removeApplicationToSoftkeyTemplate	❌	❌	❌	✅	❌	❌	❌	✅	✅	
updateAppUser	❌	❌	❌	❌	✅	✅	✅	ℹ️	✅	
addAttendantConsoleHuntGroup	❌	❌	❌	✅	✅	✅	✅	✅	✅	
removeAttendantConsoleHuntGroup	❌	❌	❌	✅	✅	✅	✅	✅	✅	
updateAttendantConsoleHuntGroup	❌	❌	❌	✅	ℹ️	✅	✅	✅	✅	
getAttendantConsoleHuntGroup	❌	❌	❌	✅	✅	✅	✅	✅	✅	
addAttendantConsoleUser	❌	❌	❌	✅	✅	✅	✅	✅	✅	
removeAttendantConsoleUser	❌	❌	❌	✅	✅	✅	✅	✅	✅	
updateAttendantConsoleUser	❌	❌	❌	✅	✅	✅	✅	✅	✅	
getAttendantConsoleUser	❌	❌	❌	✅	✅	✅	✅	✅	✅	
addCalledPartyTransformationPattern	❌	❌	❌	❌	❌	❌	❌	❌	✅	
removeCalledPartyTransformationPattern	❌	❌	❌	❌	❌	❌	❌	❌	✅	
updateCalledPartyTransformationPattern	❌	❌	❌	❌	❌	❌	❌	❌	✅	
getCalledPartyTransformationPattern	❌	❌	❌	❌	❌	❌	❌	❌	✅	
addCallerFilterList	❌	❌	❌	❌	❌	❌	❌	✅	✅	
removeCallerFilterList	❌	❌	❌	❌	❌	❌	❌	✅	✅	
updateCallerFilterList	❌	❌	❌	❌	❌	❌	❌	✅	✅	
getCallerFilterList	❌	❌	❌	❌	❌	❌	❌	✅	✅	
addCallManager	✅	✅	✅	✅	ℹ️	✅	✅	✅	✅	
removeCallManager	✅	✅	✅	✅	✅	✅	✅	✅	✅	
updateCallManager	✅	✅	✅	✅	ℹ️	✅	✅	✅	✅	

getCallManager	✓	✓	✓	✓	ⓘ	✓	✓	✓	✓
addCallManagerGroup	✓	✓	✓	✓	✓	✓	✓	✓	✓
removeCallManagerGroup	✓	✓	✓	✓	✓	✓	✓	✓	✓
updateCallManagerGroup	✓	✓	✓	✓	✓	✓	✓	✓	✓
getCallManagerGroup	✓	✓	✓	✓	✓	✓	✓	✓	✓
addCallPark	✓	✓	✓	✓	ⓘ	✓	✓	✓	✓
removeCallPark	✓	✓	✓	✓	✓	✓	✓	✓	✓
updateCallPark	✓	✓	✓	✓	✓	✓	✓	✓	✓
getCallPark	✓	✓	✓	✓	✓	✓	✓	✓	✓
addCallPickupGroup	✓	✓	✓	✓	✓	✓	✓	ⓘ	✓
removeCallPickupGroup	✓	✓	✓	✓	✓	✓	✓	✓	✓
updateCallPickupGroup	✓	✓	✓	✓	✓	✓	✓	ⓘ	✓
getCallPickupGroup	✓	✓	✓	✓	✓	✓	✓	ⓘ	✓
getCCMVersion	✗	✗	✗	✗	✗	✗	✗	✓	✓
addCMCInfo	✗	✗	✓	✓	✓	✓	✓	✓	✓
removeCMCInfo	✗	✗	✓	✓	✓	✓	✓	✓	✓
updateCMCInfo	✗	✗	✓	✓	✓	✓	✓	✓	✓
getCMCInfo	✗	✗	✓	✓	✓	✓	✓	✓	✓
addCommonDeviceConfig	✗	✗	✗	✗	✗	✗	✗	✓	ⓘ
removeCommonDeviceConfig	✗	✗	✗	✗	✗	✗	✗	✓	✓
updateCommonDeviceConfig	✗	✗	✗	✗	✗	✗	✗	✓	ⓘ
getCommonDeviceConfig	✗	✗	✗	✗	✗	✗	✗	✓	ⓘ
addConferenceBridge	✗	✗	✗	✓	✓	✓	✓	ⓘ	ⓘ
removeConferenceBridge	✗	✗	✗	✓	✓	✓	✓	✓	✓
updateConferenceBridge	✗	✗	✗	✓	ⓘ	✓	✓	ⓘ	ⓘ
getConferenceBridge	✗	✗	✗	✓	✓	✓	✓	ⓘ	ⓘ
addCredentialPolicy	✗	✗	✗	✗	✗	✗	✗	✓	✓
removeCredentialPolicy	✗	✗	✗	✗	✗	✗	✗	✓	✓
updateCredentialPolicy	✗	✗	✗	✗	✗	✗	✗	✓	✓
getCredentialPolicy	✗	✗	✗	✗	✗	✗	✗	✓	✓
addCSS	✓	✓	✓	✓	✓	✓	✓	ⓘ	✓
removeCSS	✓	✓	✓	✓	✓	✓	✓	✓	✓
updateCSS	✓	✓	✓	✓	✓	✓	✓	ⓘ	✓
getCSS	✓	✓	✓	✓	✓	✓	✓	ⓘ	✓
listCSSByName	✓	✓	✓	✓	✓	✓	✓	✓	✓
addCTIRoutePoint	✓	ⓘ	✓	✓	✓	✓	✓	✓	ⓘ
removeCTIRoutePoint	✓	✓	✓	✓	✓	✓	✓	✓	✓
updateCTIRoutePoint	✓	ⓘ	✓	✓	✓	✓	✓	✓	ⓘ
getCTIRoutePoint	✓	ⓘ	✓	✓	✓	✓	✓	✓	ⓘ
addDDI	✓	✓	✓	✓	✗	✗	✗	✗	✗

Use IDP to add DDI, DialPlan and DialPlanTag

removeDDI		Use IDP to remove DDI, DialPlan and DialPlanTag
updateDDI		Use IDP to update DDI, DialPlan and DialPlanTag
getDDI		
addDeviceMobility		
removeDeviceMobility		
updateDeviceMobility		
getDeviceMobility		
addDeviceMobilityGroup		
removeDeviceMobilityGroup		
updateDeviceMobilityGroup		
getDeviceMobilityGroup		
addDeviceProfile		
removeDeviceProfile		
updateDeviceProfile		
getDeviceProfile		
addDevicePool		
removeDevicePool		
updateDevicePool		
getDevicePool		
listDevicePoolByName		
addDialPlan		Use IDP to add DDI, DialPlan and DialPlanTag
removeDialPlan		Use IDP to remove DDI, DialPlan and DialPlanTag
updateDialPlan		Use IDP to update DDI, DialPlan and DialPlanTag
getDialPlan		
addDialPlanTag		Use IDP to add DDI, DialPlan and DialPlanTag
removeDialPlanTag		Use IDP to remove DDI, DialPlan and DialPlanTag
updateDialPlanTag		Use IDP to update DDI, DialPlan and DialPlanTag
getDialPlanTag		
addDirectedCallPark		
removeDirectedCallPark		
updateDirectedCallPark		
getDirectedCallPark		
addFACInfo		
removeFACInfo		

updateFACInfo
getFACInfo
addGatekeeper
removeGatekeeper
updateGatekeeper
getGatekeeper
listGatekeeperByName
addGatewayEndpoint
removeGatewayEndpoint
updateGatewayEndpoint
getGatewayEndpoint
addH323Gateway
removeH323Gateway
updateH323Gateway
getH323Gateway
addH323Phone
removeH323Phone
updateH323Phone
getH323Phone
addH323Trunk
removeH323Trunk
updateH323Trunk
getH323Trunk
addHuntList
removeHuntList
updateHuntList
getHuntList
addHuntPilot
removeHuntPilot
updateHuntPilot
getHuntPilot
addIVRUserLocale
removeIVRUserLocale
aupdateIVRUserLocale
agetIVRUserLocale
updateLicenseCapabilities
getLicenseCapabilities
addLine
removeLine
updateLine

getLine		
addLineGroup		
removeLineGroup		
updateLineGroup		
getLineGroup		
addLocation		
removeLocation		
updateLocation		
getLocation		
listLocationByName		
addMediaResourceGroup		
removeMediaResourceGroup		
updateMediaResourceGroup		
getMediaResourceGroup		
listMediaResourceGroupByName		
addMediaResourceList		
removeMediaResourceList		
updateMediaResourceList		
getMediaResourceList		
addMeetMe		
removeMeetMe		
updateMeetMe		
getMeetMe		
listMediaResourceListByName		
addMGCP		MGCP is the box level configuration for a gateway.
removeMGCP		
updateMGCP		
getMGCP		
addMGCPEndpoint		MGCPEndpoint is the port on the gateway.
removeMGCPEndpoint		
addMGCPUnit		MGCPUnit is the gateway Network Module.
removeMGCPUnit		
addMGCPSubunit		MGCPSubunit is the gateway VIC or VWIC.
removeMGCPSubunit		
addMobileVoiceAccess		
removeMobileVoiceAccess		
updateMobileVoiceAccess		

getMobileVoiceAccess		
addMobility		
removeMobility		
updateMobility		
getMobility		
updateMOHAudioSource		
removeMOHAudioSource		Blanks out the MOHAudioSource as if it were removed.
getMOHAudioSource		
listMOHAudioSourceByName		
addMOHServer		
removeMOHServer		
updateMOHServer		
getMOHServer		
addPhone		
removePhone		
updatePhone		
getPhone		
listPhoneByDescription		
listPhoneByName		
listPhoneTemplateByName		
addPhoneTemplate		
removePhoneTemplate		
updatePhoneTemplate		
getPhoneTemplate		
addPhysicalLocation		
removePhysicalLocation		
updatePhysicalLocation		
getPhysicalLocation		
addPilotPoint		
removePilotPoint		
updatePilotPoint		
getPilotPoint		
addProcessNode		
removeProcessNode		
updateProcessNode		
getProcessNode		
updateProcessNodeService		
getProcessNodeService		
listProcessNodesByService		
listAllProcessNodes		

addRecordingProfile
removeRecordingProfile
updateRecordingProfile
getRecordingProfile
addRegion
removeRegion
updateRegion
getRegion
updateRegionMatrix
addRemoteDestination
removeRemoteDestination
updateRemoteDestination
getRemoteDestination
addRemoteDestinationProfile
removeRemoteDestinationProfile
updateRemoteDestinationProfile
getRemoteDestinationProfile
updateResourcePriorityDefaultNamespace
getResourcePriorityDefaultNamespace
addResourcePriorityNamespace
removeResourcePriorityNamespace
updateResourcePriorityNamespace
getResourcePriorityNamespace
addResourcePriorityNamespaceList
removeResourcePriorityNamespaceList
updateResourcePriorityNamespaceList
getResourcePriorityNamespaceList
addRouteFilter
removeRouteFilter
updateRouteFilter
getRouteFilter
addRouteGroup
removeRouteGroup
updateRouteGroup
getRouteGroup
addRouteList
removeRouteList
updateRouteList
getRouteList
addRoutePartition

removeRoutePartition
updateRoutePartition
getRoutePartition
listRoutePartitionByName
addRoutePattern
removeRoutePattern
updateRoutePattern
getRoutePattern
listRoutePlanByType
updateServiceParameter
getServiceParameter
listServiceParameters
addSIPProfile
removeSIPProfile
updateSIPProfile
getSIPProfile
addSIPRealm
removeSIPRealm
updateSIPRealm
getSIPRealm
addSIPTrunk
removeSIPTrunk
updateSIPTrunk
getSIPTrunk
updateSoftKeySet
getSoftKeySet
addSoftKeyTemplate
removeSoftKeyTemplate
updateSoftKeyTemplate
getSoftKeyTemplate
addTimePeriod
removeTimePeriod
updateTimePeriod
getTimePeriod
addTimeSchedule
removeTimeSchedule
updateTimeSchedule
getTimeSchedule
addTODAccess
removeTODAccess

updateTODAccess		
getTODAccess		
addTranscoder		
removeTranscoder		
updateTranscoder		
getTranscoder		
addTransformationPattern		
removeTransformationPattern		
updateTransformationPattern		
getTransformationPattern		
addTransPattern		for adding CallingPartyTransformationPattern
removeTransPattern		for removing CallingPartyTransformationPattern
updateTransPattern		for udpating CallingPartyTransformationPattern
getTransPattern		for getting CallingPartyTransformationPattern
addUser		
removeUser		
updateUser		
getUser		
listUserByName		
addUserGroup		
removeUserGroup		
updateUserGroup		
getUserGroup		
addVoiceMailPilot		
removeVoiceMailPilot		
updateVoiceMailPilot		
getVoiceMailPilot		
addVoiceMailPort		
removeVoiceMailPort		
updateVoiceMailPort		
getVoiceMailPort		
addVoiceMailProfile		
removeVoiceMailProfile		
updateVoiceMailProfile		
getVoiceMailProfile		
listVoiceMailProfileByName		
createAutogeneratedProfile		

doAuthenticateUser	✗	✗	✗	✗	✓	✓	✓	✓	✓
doDeviceLogin	✓	✓	✓	✓	✓	✓	✓	✓	✓
doDeviceLogout	✓	✓	✓	✓	ℹ	✓	✓	✓	✓
doDeviceReset	✓	✓	✓	✓	ℹ	✓	✓	✓	✓
executeSQLQuery	✗	✓	✓	✓	✓	✓	✓	✓	✓
executeSQLUpdate	✗	✗	✗	✗	✓	✓	✓	✓	✓
getNumDevices	✓	✓	✓	✓	✓	✓	✓	✓	✓
listDeviceByNameAndClass	✓	✓	✓	✓	✓	✓	✓	✓	✓

INDEX

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

W