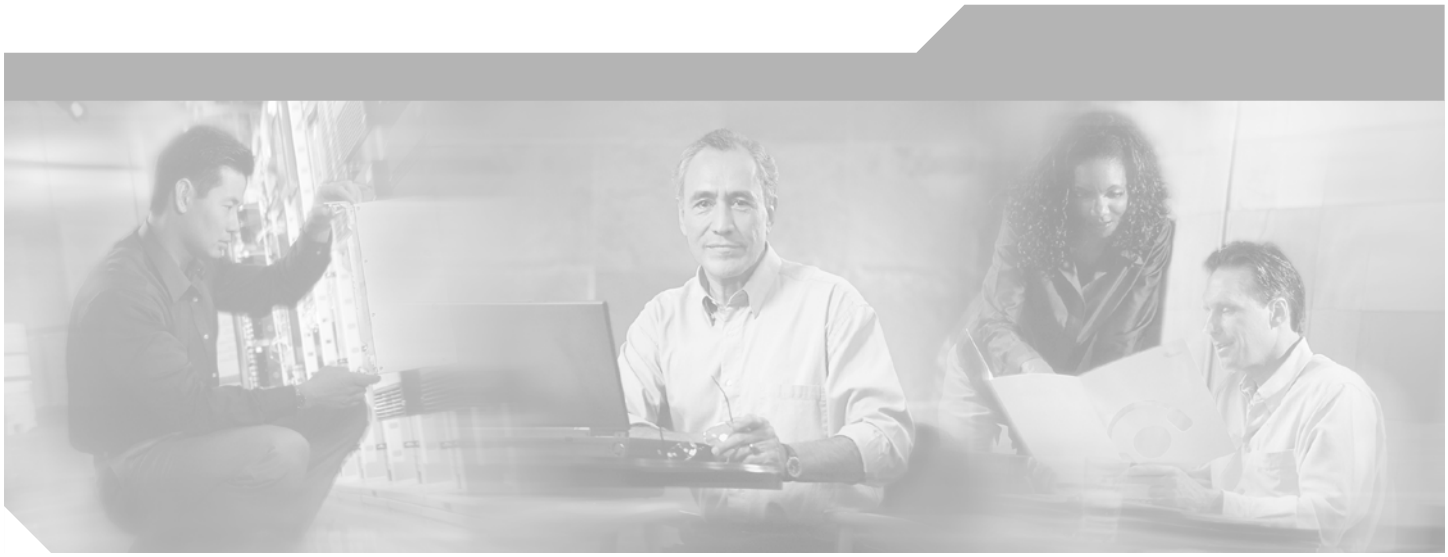


Cisco Unified CallManager Developers Guide for Release 5.0

Corporate 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: 408 526-4100

Text Part Number: OL-9435-01

CONTENTS

INDEX

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

The purpose of this document is to describe the following Cisco Unified CallManager APIs:

- The Cisco Unified CallManager AXL implementation allows applications to modify the Cisco Unified CallManager system database. AXL is not intended as a real-time API, but as a provisioning and configuration API.
- Cisco Unified CallManager Real-Time information, Performance Counters, and Database information exposure occurs through the AXL Serviceability API.
- The Cisco Unified CallManager 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 CiscoUnified CallManager 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 CiscoUnified CallManager Extension Mobility Service make many other applications possible.

  Some examples are:

  - 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 CallManager WebDialer application is installed on a Cisco Unified CallManager server. It 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 on the telephone number of the person they are trying to call.

Audience

The *Cisco Unified CallManager Developers Guide* provides information for developers who write applications that extend the functionality of the APIs 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

In addition, users of the Cisco Unified CallManager 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 CallManager and its applications. Documents on Cisco Unified CallManager and other related technologies are listed in the Related Documentation section.

Organization

The following organization applies for this guide.

Table 1 Organization

Chapter	Description
Chapter 1, "AXL Programming"	This chapter describes the Administrative XML Layer (AXL) API, which provides a mechanism for inserting, retrieving, updating, and removing data from the database using an XML SOAP interface. This allows you to access Cisco Unified CallManager data 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 the SOAPISAPI.dll. Cisco Unified CallManager Real-Time information, Performance Counters, and Database information exposure occurs through the AXL Serviceability APIs.
Chapter 3, "Extension Mobility Service API Programming"	This chapter includes high-level concepts that are important in understanding the Cisco Unified CallManager Extension Mobility Service as well as an overview of configuring EM services, messages, message DTDs, and error codes.
Chapter 4, "WebDialer API Programming"	This chapter describes the Simple Object Access Protocol (SOAP) and HTML over HTTP (and HTTPS) interfaces that are used to develop customized directory search applications for Cisco Unified CallManager WebDialer.

New and Changed Information

New features and or changes that are pertinent to release 5.0 of the Cisco Unified CallManager are described in each chapter,

Related Documentation

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

- Cisco Unified CallManager Release 5.0—A suite of documents related to the installation and configuration of Cisco Unified CallManager. Refer to the *Cisco Unified CallManager Documentation Guide for Release* 5.0 for a list of documents on installing and configuring Cisco Unified CallManager 5.0, including:
  - *Cisco Unified CallManager Administration Guide, Release* 5.0.
  - *Cisco Unified CallManager System Guide, Release* 5.0.
  - *Cisco Unified CallManager Features and Services Guide, Release* 5.0.
- *Cisco Unified IP Phones and Services*—A suite of documents related to the installation and configuration of Cisco Unified IP Phones.

- *Cisco DistributedDirector*—A suite of documents that are related 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.
^	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

Cisco documentation and additional literature are available on Cisco.com. Cisco also provides several ways to obtain technical assistance and other technical resources. These sections explain how to obtain technical information from Cisco Systems.

Cisco.com

You can access the most current Cisco documentation at this URL:

http://www.cisco.com/univercd/home/home.htm

You can access the Cisco website at this URL:

http://www.cisco.com

You can access international Cisco websites at this URL:

http://www.cisco.com/public/countries_languages.shtml

Documentation

You can find instructions for ordering documentation at this URL:

http://www.cisco.com/univercd/cc/td/doc/es_inpck/pdi.htm

You can order Cisco documentation in these ways:

- Registered Cisco.com users (Cisco direct customers) can order Cisco product documentation from the Ordering tool:

  http://www.cisco.com/en/US/partner/ordering/index.shtml

- Nonregistered Cisco.com users can order documentation through a local account representative by calling Cisco Systems Corporate Headquarters (California, USA) at 408 526-7208 or, elsewhere in North America, by calling 800 553-NETS (6387).

Documentation Feedback

You can send comments about technical documentation to bug-doc@cisco.com.

You can submit comments by using the response card (if present) behind the front cover of your document or by writing to the following address:

Cisco Systems
Attn: Customer Document Ordering
170 West Tasman Drive
San Jose, CA 95134-9883

We appreciate your comments.

Cisco Product Security Overview

Cisco provides a free online Security Vulnerability Policy portal at this URL:

http://www.cisco.com/en/US/products/products_security_vulnerability_policy.html

From this site, you can perform these tasks:

- Report security vulnerabilities in Cisco products.
- Obtain assistance with security incidents that involve Cisco products.
- Register to receive security information from Cisco.

A current list of security advisories and notices for Cisco products is available at this URL:

http://www.cisco.com/go/psirt

If you prefer to see advisories and notices as they are updated in real time, you can access a Product Security Incident Response Team Really Simple Syndication (PSIRT RSS) feed from this URL:

http://www.cisco.com/en/US/products/products_psirt_rss_feed.html

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.

Reporting Security Problems in Cisco Products

Cisco is committed to delivering secure products. We test our products internally before we release them, and we strive to correct all vulnerabilities quickly. If you think that you might have identified a vulnerability in a Cisco product, contact PSIRT:

- Emergencies—security-alert@cisco.com
- Nonemergencies—psirt@cisco.com

Tip We encourage you to use Pretty Good Privacy (PGP) or a compatible product to encrypt any sensitive information that you send to Cisco. PSIRT can work from encrypted information that is compatible with PGP versions 2.*x* through 8.*x*.

Never use a revoked or an expired encryption key. The correct public key to use in your correspondence with PSIRT is the one that has the most recent creation date in this public key server list:

http://pgp.mit.edu:11371/pks/lookup?search=psirt%40cisco.com&op=index&exact=on

In an emergency, you can also reach PSIRT by telephone:

- 1 877 228-7302
- 1 408 525-6532

Obtaining Technical Assistance

For all customers, partners, resellers, and distributors who hold valid Cisco service contracts, Cisco Technical Support provides 24-hour-a-day, award-winning technical assistance. The Cisco Technical Support Website on Cisco.com features extensive online support resources. In addition, Cisco Technical Assistance Center (TAC) engineers provide telephone support. If you do not hold a valid Cisco service contract, contact your reseller.

Cisco Technical Support Website

The Cisco Technical Support Website provides online documents and tools for troubleshooting and resolving technical issues with Cisco products and technologies. The website is available 24 hours a day, 365 days a year at this URL:

http://www.cisco.com/techsupport

Access to all tools on the Cisco Technical Support Website requires a Cisco.com user ID and password. If you have a valid service contract but do not have a user ID or password, you can register at this URL:

http://tools.cisco.com/RPF/register/register.do

Developer Support

The Developer Support Program provides formalized support for Cisco Systems interfaces to enable developers, customers, and partners in the Cisco Service Provider Solutions Ecosystem and Cisco Partner programs to accelerate their delivery of compatible solutions.

The Developer Support Engineers are an extension of the product technology engineering teams. They have direct access to the resources necessary to provide expert support in a timely manner.

For additional information on this program, refer to the Developer Support Program Web Site at http://www.cisco.com/go/developer support/.

Developers using the *Cisco Unified CallManager Developers Guide* are encouraged to join the Cisco Developer Support Program. This program provides a consistent level of support while leveraging Cisco interfaces in development projects.

Note Cisco Technical Assistance Center (TAC) support does not include Extension Mobility API developer support and is limited to Cisco Unified Communications installation/configuration and Cisco-developed applications. For more information about the Developer Support Program, please contact Cisco at developer-support@cisco.com.

Submitting a Service Request

Using the online TAC Service Request Tool is the fastest way to open S3 and S4 service requests. (S3 and S4 service requests are those in which your network is minimally impaired or for which you require product information.) After you describe your situation, the TAC Service Request Tool automatically provides recommended solutions. If your issue is not resolved using the recommended resources, your service request will be assigned to a Cisco TAC engineer. The TAC Service Request Tool is located at this URL:

http://www.cisco.com/techsupport/servicerequest

For S1 or S2 service requests or if you do not have Internet access, contact the Cisco TAC by telephone. (S1 or S2 service requests are those in which your production network is down or severely degraded.) Cisco TAC engineers are assigned immediately to S1 and S2 service requests to help keep your business operations running smoothly.

To open a service request by telephone, use one of the following numbers:

Asia-Pacific: +61 2 8446 7411 (Australia: 1 800 805 227)
EMEA: +32 2 704 55 55
USA: 1 800 553 2447

For a complete list of Cisco TAC contacts, go to this URL:

http://www.cisco.com/techsupport/contacts

Definitions of Service Request Severity

To ensure that all service requests are reported in a standard format, Cisco has established severity definitions.

Severity 1 (S1)—Your network is “down,” or there is a critical impact to your business operations. You and Cisco will commit all necessary resources around the clock to resolve the situation.

Severity 2 (S2)—Operation of an existing network is severely degraded, or significant aspects of your business operation are negatively affected by inadequate performance of Cisco products. You and Cisco will commit full-time resources during normal business hours to resolve the situation.

Severity 3 (S3)—Operational performance of your network is impaired, but most business operations remain functional. You and Cisco will commit resources during normal business hours to restore service to satisfactory levels.

Severity 4 (S4)—You require information or assistance with Cisco product capabilities, installation, or configuration. There is little or no effect on your business operations.

Obtaining Additional Publications and Information

Information about Cisco products, technologies, and network solutions is available from various online and printed sources.

- Cisco Marketplace provides a variety of Cisco books, reference guides, and logo merchandise. Visit Cisco Marketplace, the company store, at this URL:

  http://www.cisco.com/go/marketplace/

- The Cisco *Product Catalog* describes the networking products offered by Cisco Systems, as well as ordering and customer support services. Access the Cisco Product Catalog at this URL:

  http://cisco.com/univercd/cc/td/doc/pcat/

- *Cisco Press* publishes a wide range of general networking, training and certification titles. Both new and experienced users will benefit from these publications. For current Cisco Press titles and other information, go to Cisco Press at this URL:

  http://www.ciscopress.com

- *Packet* magazine is the Cisco Systems technical user magazine for maximizing Internet and networking investments. Each quarter, Packet delivers coverage of the latest industry trends, technology breakthroughs, and Cisco products and solutions, as well as network deployment and troubleshooting tips, configuration examples, customer case studies, certification and training information, and links to scores of in-depth online resources. You can access Packet magazine at this URL:

  http://www.cisco.com/packet

- *iQ Magazine* is the quarterly publication from Cisco Systems designed to help growing companies learn how they can use technology to increase revenue, streamline their business, and expand services. The publication identifies the challenges facing these companies and the technologies to help solve them, using real-world case studies and business strategies to help readers make sound technology investment decisions. You can access iQ Magazine at this URL:

  http://www.cisco.com/go/iqmagazine

- *Internet Protocol Journal* is a quarterly journal published by Cisco Systems for engineering professionals involved in designing, developing, and operating public and private internets and intranets. You can access the Internet Protocol Journal at this URL:

  http://www.cisco.com/ipj

- World-class networking training is available from Cisco. You can view current offerings at this URL:

  http://www.cisco.com/en/US/learning/index.html

AXL Programming

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

This chapter contains the following sections:

Introduction

The Administrative XML Layer (AXL) Application Programming Interface (API) provides a mechanism for inserting, retrieving, updating, and removing data from the database by using an eXtensible Markup Language (XML) Simple Object Access Protocol (SOAP) interface. This allows a programmer to access Cisco Unified CallManager data 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, are performed using a combination of HTTP and SOAP. SOAP is an XML remote procedure call (RPC) protocol. Users perform requests by sending XML data to the Cisco Unified CallManager server. The server then returns the AXL response, which is also a SOAP message.

Target Audience for this Chapter

This chapter targets somewhat experienced developers who would like access to one or more of the following items:

- Cisco Unified CallManager data
- Cisco Unified CallManager data in XML format
- Cisco Unified CallManager data in a platform-independent manner

This chapter 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:

- TCP/IP Protocol
- Hypertext Transport Protocol
- Socket programming
- XML

In addition, users of the AXL API must have a firm grasp of XML 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/.

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

New and Changed Information

The following list provides AXL API calls that are new in Release 5.0:

- executeSQLUpdate
- doAuthenticateUser
- updateAppUser
- addUserGroup
- updateUserGroup
- removeUserGroup
- getUserGroup

The following list gives AXL API calls that have been changed in Release 5.0:

- addPhone
- updatePhone
- getPhone
- addGatewayEndpoint
- updateGatewayEndpoint
- getGatewayEndpoint

- addMGCPEndpoint
- updateMGCP
- addSIPTrunk
- updateSIPTrunk
- getSIPTrunk
- addCallManager
- updateCallManager
- getCallManager
- addCallPark
- addRoutePattern
- updateRoutePattern
- updateTransPattern
- updateHuntPilot
- addHuntList
- updateHuntList
- getHuntList
- addPilotPoint
- updatePilotPoint
- getPilotPoint
- addH323Gateway
- updateH323Gateway
- updateH323Phone
- getH323Gateway
- getH323Trunk
- addUser
- updateUser
- getUser
- updateProcessNodeService
- getProcessNodeService
- doDeviceLogout
- listUserByName
- updateServiceParameter
- updateGatekeeper
- updateConferenceBridge
- updateAttendantConsoleHuntGroup
- updateDeviceProfile
- updateLine
- updateLineGroup

- addDevicePool
- updateDevicePool
- doDeviceReset

The following list gives API calls that have been deprecated in Release 5.0:

- addDDI
- updateDDI
- removeDDI
- addDialPlan
- updateDialPlan
- removeDialPlan
- addDialPlanTag
- updateDialPlanTag
- removeDialPlanTag

AXL API

Request methods are 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, then the appropriate AXL response is returned. All responses are named identically to the associated requests, except that the word "Response" is appended.

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

If an error occurs, an XML error structure is returned wrapped inside a SOAP Fault structure (see the "AXL Error Codes" section on page 1-4).

AXL Compliance

The Cisco Unified CallManager AXL implemetation complies with XML Schema 1.0, which was tested for XML Schema compliance with a third party application 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 CallManager AXL implemetation 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 only be accessed 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, an error is returned 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 is an example of a detailed SOAP Fault.

```
<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/CMApi/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 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> elements:ws The user agent uses the <code> element, a numerical value, to find out what type of error occurred. The error codes follow:

Error Code	Description
Less than 5000	These errors directly correspond to DBL Exception error codes. Refer to the documentation for the DBLException class for explanations of these errors.
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.
5001	**Parser Error**—An error occurred while parsing the XML request.
5002	**Unknown Request Error**—The user agent has submitted a request that is unknown to the API.
5003	**Invalid Value Exception**—An invalid value is detected in the XML request.

Error Code	Description
5004	**AXL Unavailable Exception**—The AXL service is too busy to handle the request at this time. The request should be sent again at a later time.
5005	**Unexpected Node Exception**—The server encountered an unexpected element. For example, if the server expects the next node to be <name>, but encounters <protocol>, this error is returned. Malformed requests that do not adhere to the latest AXL Schema will cause these errors.
5007	**Item Not Valid Error**—The specified item is 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 what type of request generated this error. Because this element is optional, it may not always appear.

Examples of AXL Requests

There are no platform considerations in Cisco Unified CallManager Release 5.0. 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 CallManger 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, an HTTP 415 error is returned 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, an HTTP 401 Access Denied error is returned 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, an HTTP 413 error message is returned.

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

```
POST :8443/axl
Host: axl.myhost.com:80
Accept: text/*
Authorization: Basic bGFycnk6Y3VybHkgYW5kIG1vZQ==
Content-type: text/xml
Content-length: 613
```

The following AXL request is used in the code examples that are displayed in the following sections. This example shows a getPhone request:

```
POST :8443/axl
Host: axl.myhost.com:80
Accept: text/*
Authorization: Basic bGFycnk6Y3VybHkgYW5kIG1vZQ==
Content-type: text/xml
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 <winsock2.h>           // required for sockets
#include <iostream>             // required for console I/O
#include <sstream>
#include <string>               // required for std::string

using namespace std;

int main(int argc, char* argv[])
{

// make connection to server

    WSADATA WSAData;

    // initialize sockets
    if (int iError = WSAStartup (MAKEWORD(2,0), &WSAData))
{
        cout << "Windows Sockets startup error. Aborting." << endl;
        return -1;
}

SOCKET Socket = socket (AF_INET, SOCK_STREAM, IPPROTO_TCP);
```

```
if (Socket == INVALID_SOCKET)
{
    cout << "Socket creation error. Aborting." << endl;
    return -1;
}

SOCKADDR_IN sinRemote;

sinRemote.sin_family = AF_INET;
sinRemote.sin_port = htons (80) ;
sinRemote.sin_addr.s_addr =  inet_addr( "127.0.0.1" );

cout << "connecting to service" << endl;
int retval = connect(Socket, (SOCKADDR *)&sinRemote, sizeof (sinRemote));

if (retval != 0)
{
        cout << "Error occured while connecting to socket. Aborting." << endl;
        closesocket(Socket);
        return -1;
}

const int BUFSIZE = 2048;
char buff[BUFSIZE];           // the temporary receive buffer
string strHTTPHeader;         // the HTTP Header
string strAXLRequest;         // the AXL SOAP request
// The AXL request:  getPhone
strAXLRequest = "<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>";

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

// build the HTTPS header
strHTTPHeader = "POST :8443/axl\r\n \
Host: localhost:80\r\n \

Authorization: Basic bGFycnk6Y3VybHkgYW5kIG1vZQ==\r\n \
Accept: text/*\r\n \
Content-type: text/xml\r\n \
Content-length: ";

strHTTPHeader += buff;
strHTTPHeader += "\r\n\r\n";

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

// send these bytes to the socket
retval = send (Socket, strAXLRequest.c_str(),strAXLRequest.length(), 0);
if ( retval != SOCKET_ERROR)
{
```

```
// output response
    cout << "received response: " << endl;
    int iTotalRead = 0;

// read BUFSIZE at a time, writing to another ostringstream
do {
    iNumRead = recv (Socket, buff, BUFSIZE-1, 0);
    buff[iNumRead] = NULL;
    cout << buff;
    iTotalRead += iNumRead;
    } while (iNumRead == BUFSIZE-1);

    cout << "Read " << iTotalRead << " bytes." << endl;
}
else
{
    cout << "An error occured while sending the data to socket." << endl;
}

// all finished, close socket
closesocket(Socket);

return 0;
```

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.*;

public class main
{
public static void main(String[] args)
{
    //Declare references
    String sAXLSOAPRequest = null;     // will hold the complete request,
                                       // HTTPS header and SOAP payload
    String sAXLRequest = null;            // will hold only the SOAP payload
    Socket socket = null;                 // socket to AXL server
    OutputStream out = null;              // output stream to server
    InputStream in = null;                // input stream from server
    byte[] bArray = null;                 // buffer for reading response from server

    // Build the HTTPS Header
    sAXLSOAPRequest = "POST :8443/axl\r\n";
    sAXLSOAPRequest += "Host: localhost:80\r\n";
    sAXLSOAPRequest += "Authorization: Basic bGFycnk6Y3VybHkgYW5kIG1vZQ==\r\n";
    sAXLSOAPRequest += "Accept: text/*\r\n";
    sAXLSOAPRequest += "Content-type: text/xml\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>SEP222222222245</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;

    // now that the message has been built, we can connect to server and send it
       try
       {
              socket = new Socket("localhost", 80);
              out = socket.getOutputStream();
              in = socket.getInputStream();

              // send the request to the host
              out.write(sAXLSOAPRequest.getBytes());

              // read the response from the host
              StringBuffer sb = new StringBuffer(2048);
              bArray = new byte[2048];
              int ch = 0;
              int sum = 0;
              while ( (ch = in.read(bArray)) != -1 )
              {
                     sum += ch;
                     sb.append(new String(bArray, 0, ch));
              }

              socket.close();

              // output the response to the standard out
              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
              try
              {
                     if (socket != null) socket.close();
              } catch (Exception exc)
              {
                     System.err.println("Error closing connection to server: " +
exc.getMessage());
              }
              return;
       }
```

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

Throttling of Requests

The side-effects of updating the Cisco Unified CallManager 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 using the Database Layer service parameter “MaxAXLWritesPerMinute.”

AXL accommodates all requests until the "MaxAXLWritesPerMinute" value is reached. Subsequent attempts to modify the database with AXL are rejected with an HTTPS 503 Service Unavailable response. Every minute, AXL resets its internal counter and begins to accept AXL update requests until the limit gets reached again.

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

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

The AXL Schema Documentation

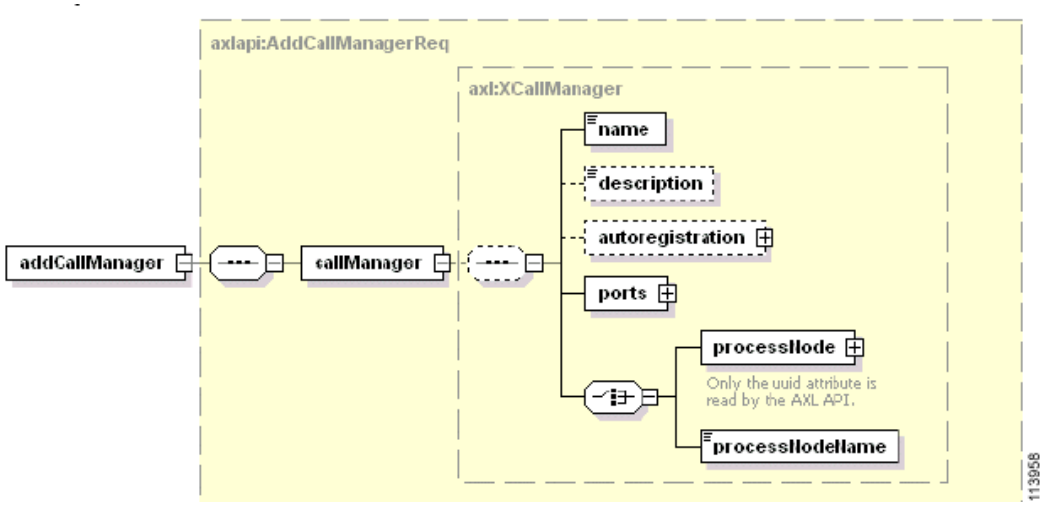
The axlsqltoolkit.zip plugin contains the following six AXL schema files: AXLAPI.wsdl, AXLEnums.xsd, axlmessage.xsd, axlsoap.xsd, axl.xsd, and SoapEnvelope.xsd. These files encapsulate the complete AXL schema (including details of all requests, responses, XML objects, data types, and so on).

The AXL schema files can also be obtained through the Cisco developer support program at http://www.cisco.com/pcgi-bin/dev_support/access_level/product_support.

Standard XML handling IDEs and development environments can illuminate or auto-generate 'friendly' formatted documents based on the AXL schema files.

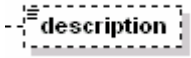
The following example describes a complete auto-generated HTML document based on the schema.

Example XML Structure



The AXL schema, which is provided as an HTML document, graphically describes each request and response. (See http://www.cisco.com/pcgi-bin/dev_support/access_level/product_support.)

The following legend explains the graphics that are used in the AXL schema document.

Request or Response	Element Name	Description
	Complex Element	A complex element can have child elements, as well as attributes. An element with a solid border must appear in an XML instance document.
	Simple Element	A simple element cannot have child elements but can have attributes. An element with a solid border must appear in an XML instance document.
	Optional Element	An optional element does not have to appear in an instance of the XML. Any type of element can be optional, including sequence and choice elements.
	Sequence Element	A sequence element means that all children of this element must appear in the XML in the order that they are listed.
	Choice Element	A choice element means that only one of the children of this element can appear in the XML.

Authentication

Deactivate anonymous access to the AXL SOAP service to enforce user authentication. User authentication gets controlled via the HTTPS Basic Authentication scheme; therefore, you must include the Authorization header in the HTTPS header.

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==” is the Base64 encoding of “larry:curly and moe.”

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

Data Encryption

Encrypt AXL SOAP messages by using HTTP SSL. SSL is functional on the web server by default. 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 CallManager 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 is a provisioning and configuration API; it is not intended to be used as a real-time API.

If AXL is determined to be using too much CPU time, consider lowering the MaxAXLWritesPerMinute service parameter. If this does not solve the problem, consider purchasing a second server to be used only by applications that are using AXL.

Note The autologout function of Extension Mobility does not work when the user is logged in/out of EM via the AXL interface. The AXL interface is not intended to be used as a real-time API.

AXL Serviceability API Programming

This chapter describes the implementation of AXL-Serviceability APIs . Cisco Unified CallManager 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 found in this document, refer to http://www.cisco.com/pcgi-bin/dev_support/access_level/product_support. You must have a Cisco CCO account and password to access this URL.

This chapter covers the following topics:

Introduction

By exposing Cisco Unified CallManager 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 CallManager Serviceability product.

AXL-Serviceability APIs provides 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 CallManager Serviceability functionalities through an open and standard transport protocol and data model.

The following enhancements exist in Serviceability for AXL in Cisco Unified CallManager Release 5.0:

- You can access AXL using Serviceability **(Serviceability > Tools AXL Web Service)**.
- Serviceability provides an option to activate/deactivate AXL Web Service **(Serviceability > Tools > Service Activation)**.
- Serviceability provides an option to start and stop AXL Web Service **(Serviceability > Tools > Control Center - Feature Services)**.
- Serviceability provides an option to change trace levels for AXL **(Serviceability > Trace > Configuration)**.

The following list gives the Serviceability services that are exposed via AXL-Serviceability APIs:

- Perfmon Data Collection

  In Cisco Unified CallManager 5.0 the Perfmon Data Collection service is exposed through a new framework called Perfmon Infrastructure and System Information Library. The APIs are the same; however, the information provided through these APIs in Cisco Unified CallManager Release 5.0 will be similar but not exactly the same. Windows performance counters are no longer available and the URL will be different from previous releases for each of the SOAP services.

- Real-Time Information (RisPort) — There are no changes in this section.

The following four services are new in Cisco Unified CallManager Release 5.0:

- Server Query Service
- Service Interface
- Log Collection Service
- CDR on Demand Service

Data Model

AXL-Serviceability APIs are based on SOAP with XML request and response messages. They must conform to the structure of a SOAP Envelope element. Although SOAP is a standard protocol, many of its data model aspects are left 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.

Downloading Serviceability SOAP WSDL Files

Cisco Unified CallManager Release 5.0 serviceability SOAP WSDL files can be downloaded from the Cisco Unified CallManager 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/service/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.

SOAP Binding

AXL-Serviceability API SOAP binding information is well specified in the binding section of the Serviceability SOAP WSDL files. Binding specifications apply to both request and response messages. SOAP Binding covers the aspects 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 sets “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 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 which 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 is:

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 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 is the operation name. The struct contains accessor element members that represent input parameters. Operations with no parameters have no members in the struct. Names of accessor elements are the same as the names of the input parameters. Values of 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 is 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 are the same as 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, contain no member elements in the call-return struct.

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

```
OperationNameResponse
```

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

The target namespace described in the "Encoding Rule" section on page 2-4, 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 2-5 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 sub-element 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 sub-element is always present as the SOAP 1.1 specification requires. The string value of the FaultString is specific only to the AXL-Serviceability APIs.

FaultActor

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

Detail

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

If AXL-Serviceability APIs do not include the detail sub-element in the fault element, the error does not relate to the request body. Data types defined in the AXL-Serviceability APIs WSDL files specify the encoding rule for the detail sub-element. The following fragments of the file describe the 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 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 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/

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

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

  This is the namespace URL for AXL-Serviceability API data types as defined in the WSDL file.

AXL-Serviceability APIs Ports

This section explains the interface of SOAP service built-in ports in detail.

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 is an example of 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 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 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 is 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 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 previously opened by the previous PerfmonOpenSession operation.
- **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 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 multi-reference 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 opened previously by the PerfmonOpenSession operation.
- **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 is an example of 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 multi reference 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 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 opened previously by the PerfmonOpenSession operation.

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 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 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 previously opened by the PerfmonOpenSession operation.

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 Cisco Unified CallManager Real-Time Information

Request Format

SOAP Action and Envelope Information

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

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

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 CallManager 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 is 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 has StateInfo as part of the response. This is the unique string that tells the client the state of the Device server information. Clients have to 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 no new registration or deregistration of devices has happened. This saves lot of time for clients and servers since the server is telling the client that there is no change from the previous state. If “NoChange” in the response is set to false, this indicates the server information has 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>
```

As part of the response a maximum of 200 devices is provided 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>
```

This is done 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% of CPU resources in call processing servers. Many clients may be requesting this information, but we have 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 first 200 device’s real-time information.

```
<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>
```

The most efficient way to get the list of devices is to 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, then iterate through the list sending 200 at a time to the AXIS-SOAP API.

Device requests per minute can not exceed 15 per minute (by default) to the Cisco Unified CallManager server. So space client requests so that 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. These rates are expected to take less than 20% of the CPU resources, which will not affect the Cisco Unified CallManager’s ability 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 CallManager SIP device related queries. HTTP header has 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 optional State Info is followed by selection criteria type CmSelectionCriteriaSIP.

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

CmSelectionCriteriaSIP is composed of the following:

- **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 is 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. No name specified will search in all servers in cluster.

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

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

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

- **SelectItems**—the array of items for which search criteria is specified. Following specifies array that contains IP Address or Device Name of the items for which we need the real time status.

```
<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 returned by the SOAP server. Each node has a sequence of search information 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 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 information defined below.

```
<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 is an example of 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.

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 is a clear-text authentication that is not secure. The Basic provides an advantage because it is widely implemented by many platforms, and most HTTP client agents support it. Basic authentication can be secure if the encryption, such as SSL, is used.

Secure

When you install/upgrade Cisco Unified CallManager, the HTTPS self-signed certificate, httpscert.cer, automatically installs on the default website that hosts the Cisco Unified CallManager 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 CallManager applications support HTTPS, which ensures the identity of the server:

- Cisco Unified CallManager Administration
- Cisco Unified CallManager Serviceability
- Cisco Unified IP Phone User Option Pages
- Cisco Unified CallManager Bulk Administration
- Cisco Unified CallManager Auto-Register Phone Tool
- Cisco Unified CallManager CDR Analysis and Reporting
- Cisco Unified CallManager Trace Collection Tool

- Cisco Unified CallManager Real Time Monitoring Tool.

For more information, refer to the *Cisco Unified CallManager Security Guide, Release* 5.0.

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, a “SOAP Fault” is sent to the client and the request is blocked. This rate control is enabled for “Real-time Data requests” and “Perfmon Data Requests.” These rates are controlled through “Enterprise parameters” as shown in Figure 2-1. The system provides Help for these parameter configurations as shown in Figure 2-2.

Figure 2-1 Rate Control Enterprise Parameters

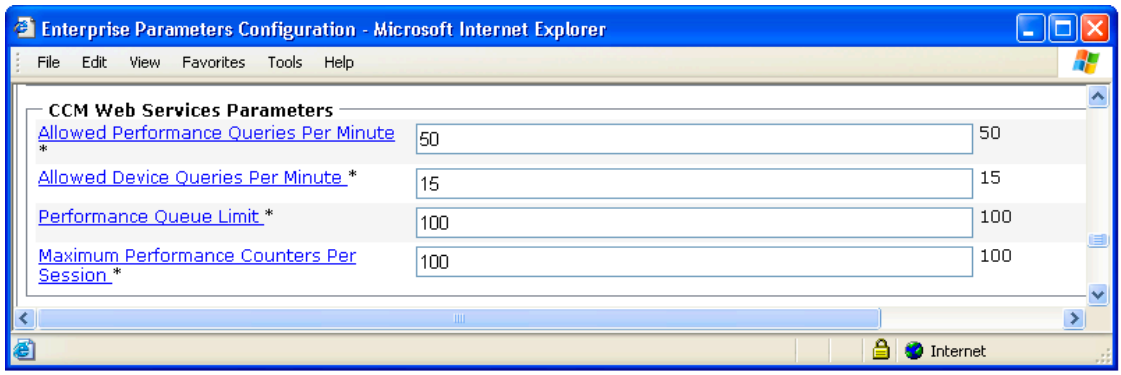
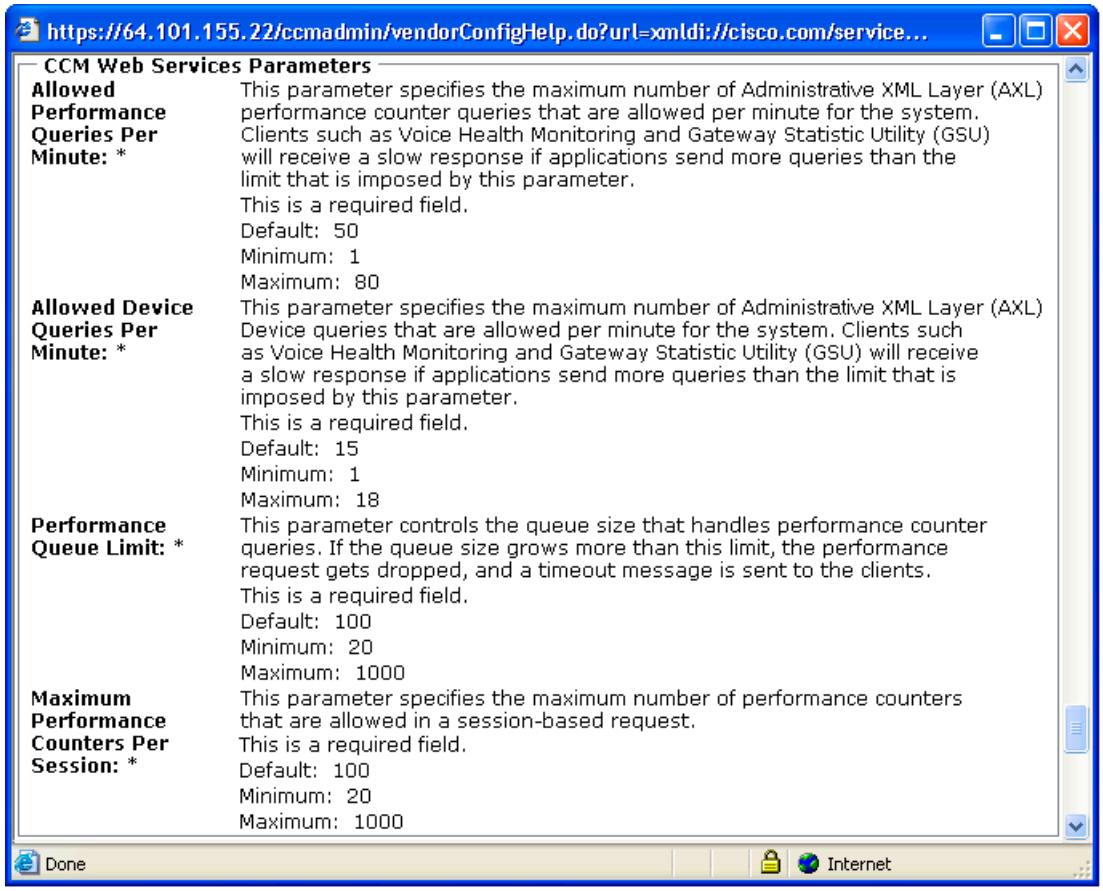


Figure 2-2 Help for Rate Control Parameters



Server Query Service

The following information is exported 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 =5.0.1.0-212

The getServerInfo operation consists of 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 is specified and the response provides the server information for the list of hostnames specified in the array.

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

Response Format

The response consists of 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 consists of 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 is 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 start and stop.

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 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 request example for getting static service specification list 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: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 response example for getting static service specification list 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: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 CallManager</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 CallManager</item>
                    </DependentServices>
                  </item>
                  <item>
                   <ServiceName xsi:type="xsd:string">Cisco Unified CallManager 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 CallManager</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 CallManager</item>
  </DependentServices>
</item>
<item>
  <ServiceName xsi:type="xsd:string">Cisco Unified CallManager 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 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 are the details about ServiceStatus, ReasonCod,e and ReasonCodeString.

- ServiceStatus is either Started or Stopped. If Started, StartTime gives the time 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 the service is Stopped:
  - If a deployable service is activated and stopped by admin, then its status is Stopped, the ReasonCode is -1019, and the corresponding ReasonCodeString is “Component is not running”.
  - If a deployable service is deactivated, then its status is Stopped, the ReasonCode is -1068, and the corresponding ReasonCodeString is “Service Not Activated”.

– If a non-deployable is stopped by admin, then its status could be Stopped with ReasonCode -1019 and the corresponding ReasonCodeString “Component is not running”.

The complete listing of ReasonCode and ReasonCodeString is as 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 non-existent service name, then ReasonCode -1010 and ReasonCodeString “No such service” will appear in the response. The following is the request and response example 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 request example for getting “Cisco Tftp” service status 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: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 response example for getting “Cisco Tftp” service status 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: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 request example for getting service status when providing an empty array of service names 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: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 above 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 CallManager</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 CallManager 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 CallManager 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 CallManager 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 CallManager 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 CallManager 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. This API can only be used 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 2-51.

Fault

Invalid Service: Non-existing Service

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

The following is the request and response example 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: Non-deployable Service

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

The following is the request and response example 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, then ReasonCode -1045 and ReasonCodeString “Operation not supported” will appear in the response.

The following is the request and response example 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 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: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 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: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 doesn't allow clients to stop the following non-stop services:

- A Red Hat DB
- Cisco Tomcat
- Cisco Database Layer Monitor
- Cisco Unified CallManager Serviceability
- Cisco SOAP -Real-Time Service APIs
- Cisco SOAP -Performace Monitoring APIs
- Cisco SOAP -Log Collection APIs

The API also doesn't allow clients to provide an empty list of services when trying to start/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 2-51.

Fault

Invalid Service: Non-existing Service

If the request of start/stop service for an invalid service name, such as a non-existing service name, then ReasonCode -1010 and ReasonCodeString "No such service" will be responsed. The following is the request and response example 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: Non-stop Service

If the request of stop service for a non-stop service, such as service “Cisco Tomcat”, then ReasonCode -1045 and ReasonCodeString “Operation not supported” will be responsed. The following is the request and response example 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 with an empty list of services, then ReasonCode -1045 and ReasonCodeString “Operation not supported” will be responsed. The following is the request and response example 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 for stop a service, and at the meantime the service status is Stopping, then ReasonCode -1045 and ReasonCodeString “Operation not supported” will be responsed. The request and response example is very similar to the example above.

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 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: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>
```

Log Collection Service

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

ListNodeServiceLogs

listNodeServiceLogs returns array of NodeServiceLogList. This array has the node names in the cluster and the list of service names / system log names.

Request Format

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

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 CallManager 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 CallManager</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 there is no node in the database, it will return a remote exception, "Query selectAllProcessNodes failed.

If there is no service list/syslog list retuned from the prefs, it will return a remote exception, No service found"/"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 CallManager 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 CallManager</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 you need to download. These are the 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 CallManager 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 CallManager</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

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 frequency specified 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 there is matching file found, it throws a remote exception. "The File Vector from the server is null”

Example

Following is the example for 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

getOneFile takes as an input parameter the absolute file name which you want to collect from the server.

The return value is the file name, but the file name will have an AXIS specific name. After the file is downloaded, you must repalce 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 is 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 is a public SOAP/HTTPS interface which is exposed to third party billing applications or customers to allow them to query the Cisco Unified CallManager 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, CDR records were stored in the CDR database and third party applications could query the database directly for the CDR records. In this release, CDRs are no longer 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 which are returned via a (s)FTP session.

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

Once 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 CallManager configuration and other criteria. If for some reason the billing servers do not receive the CDR files, or wish to have them sent again, they can do so using the SOAP/HTTPS CDR On-Demand APIs. Once CDR files are deleted, they cannot be retrieved.

The CDR On-Demand Service provides the following features:

- API to get a list of files matching a specified time interval (get_file_list)
- API to request a specific file matching a specified filename (get_file)
- Limit of 1300 file names returned from the get_file_list API
- Specified time range cannot span over one 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 the SOAP APIs must be activated from the Service Activation page 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 Click on 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 is dependent 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

Standard FTP or SFTP can be used 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 a CDR file is to be sent. Exceptions are thrown whenever an error occurs on the Servlet side. In addition, all errors will be written into log files.

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

The two APIs that comprise the CDR On-Demand Service are described in the following sections.

get_file_list

The get_file_list API allows the application to query the CDR Repository Node for a list of all the files matching a specified time interval. The time interval of the request cannot exceed one hour. If you want a list of files spanning more than the one hour time interval allowed, then you must make multiple requests to the Servlet to acquire multiple lists of filenames.

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

There is also a limit of 1300 file names that 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, then you should make additional requests with the start time of the subsequent requests as the time of the last file name 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. There is no default value.

End Time

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

Note The time span between Start Time and End Time must be a valid interval, but not longer than one 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 for a specific CDR file matching the specified filename. The resulting CDR file is then sent to the customer via standard ftp or secure ftp, depending on third party billing application's preference. The only constraint is 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 are thrown. In addition, logs will be kept detailing the errors. Log files are located 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 which 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 which 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 which 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 which the CDR Servlet is to send the CDR files.

File Name

Mandatory parameter (string) that specifies the filename of the CDR file 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 is dependent on the third party billing application configuration and preferences.

Fault

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

- Servlet is used during the maintenance period.
- The values entered for starting and ending time are not of the correct length – 12 bytes in the format YYYYMMDDHHMM is the correct length.
- The starting and ending time spans an interval of more than one hour.
- The starting time is greater than or equal to the ending time (invalid interval).

- There are no files in the CDR Repository.
- Failed to establish (s)FTP connection to the remote node.
- (s)FTP failed to send the files requested by the third party billing application.

The error condition will be described in the exception string that can be printed out by the billing application with 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>
```

Extension Mobility Service API Programming

Cisco Unified CallManager Extension Mobility (EM) service, a feature of Cisco Unified CallManager, 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 Unified CallManager Extension Mobility service works by downloading a new configuration file to the phone. Cisco Unified CallManager dynamically generates this new configuration file based on information about the user who is logging in.

Each EM user is associated 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 EM feature are authenticating the user who is logging in and generating the right configuration file based on 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 is not associated 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 is not associated 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 CallManager support for the Extension Mobility service comprises the Unified CM Extension Mobility Application (EMApp) and the Unified CM Extension Mobility Service (EMService) modules. The application and service modules, along with other Cisco Unified CallManager infrastructure such as the Database Layer (DBL), User directory (either internal or an external LDAP directory), and TFTP server provide the CiscoUnified CallManager Extension Mobility feature.

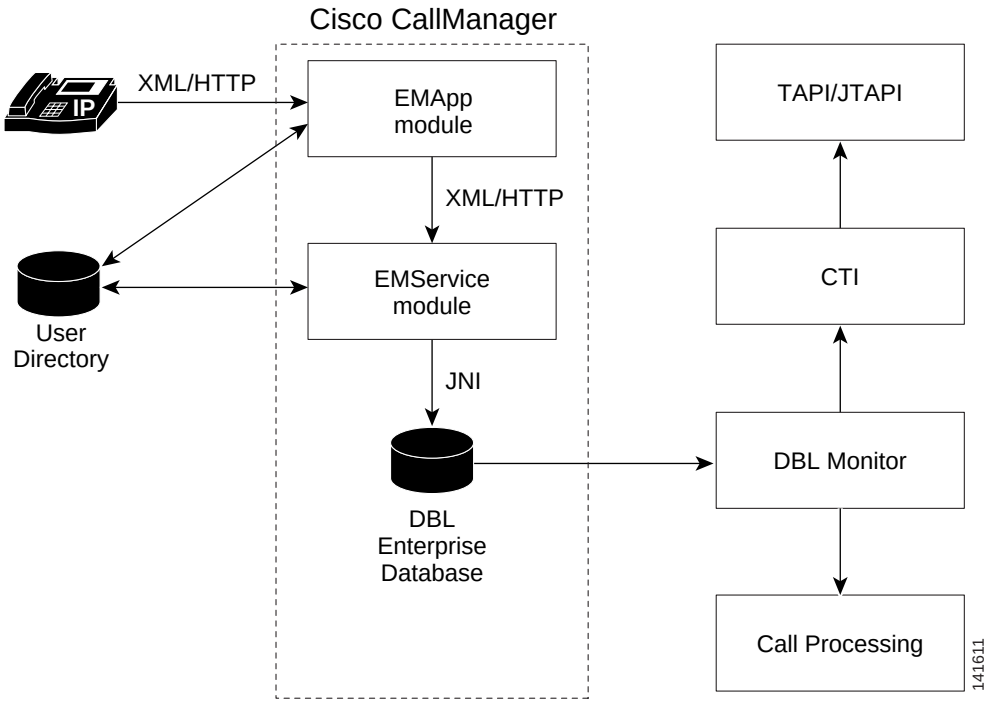
You can use the XML-based EMService API with your applications, so they can take advantage of Cisco Unified CallManager Extension Mobility service functionality. For details about how to use the Cisco Extension Mobility service API, see the "Using the Cisco Extension Mobility API" section on page 3-6.

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

Cisco Unified CallManager Extension Mobility Architecture

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

Figure 3-1 Cisco Unified CallManager Extension Mobility Functional Diagram



Cisco Unified CallManager Extension Mobility Service Components

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

- **Cisco Unified CallManager Extension Mobility Application**—A software module in Cisco Unified CallManager 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 Unified CallManager EM 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 Unified CallManager Extension Mobility Service**—A software module in Cisco Unified CallManager 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 EM 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 CallManager database and a change notification is generated to the call-processing module. This notification ultimately results in the new configuration file being pushed to the TFTP server.

The Cisco Unified CallManager 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 Unified CallManager Extension Mobility Service Components

Component	Description
DBL Monitor	Database Layer Monitor service notifies other processes of changes in the Cisco Unified CallManager database.
LDAP Directory	Lightweight Directory Access Protocol Directory (LDAP) stores information for Cisco Unified CallManager.
CallProcessing	CallProcessing is a Cisco Unified CallManager 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 Unified CallManager Extension Mobility Works

This section describes what happens when your application sends a message to the EMService to use Cisco Unified CallManager Extension Mobility functionality.

Figure 3-1 also illustrates how the Cisco Unified CallManager Extension Mobility components communicate with each other. The high-level message flow between different EM 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 Unified CallManager 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 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 Unified CallManager Extension Mobility Service Module

Your login application communicates with the Cisco Unified CallManager Extension Mobility service through the Cisco Unified CallManager Extension Mobility Service (EMService) component.

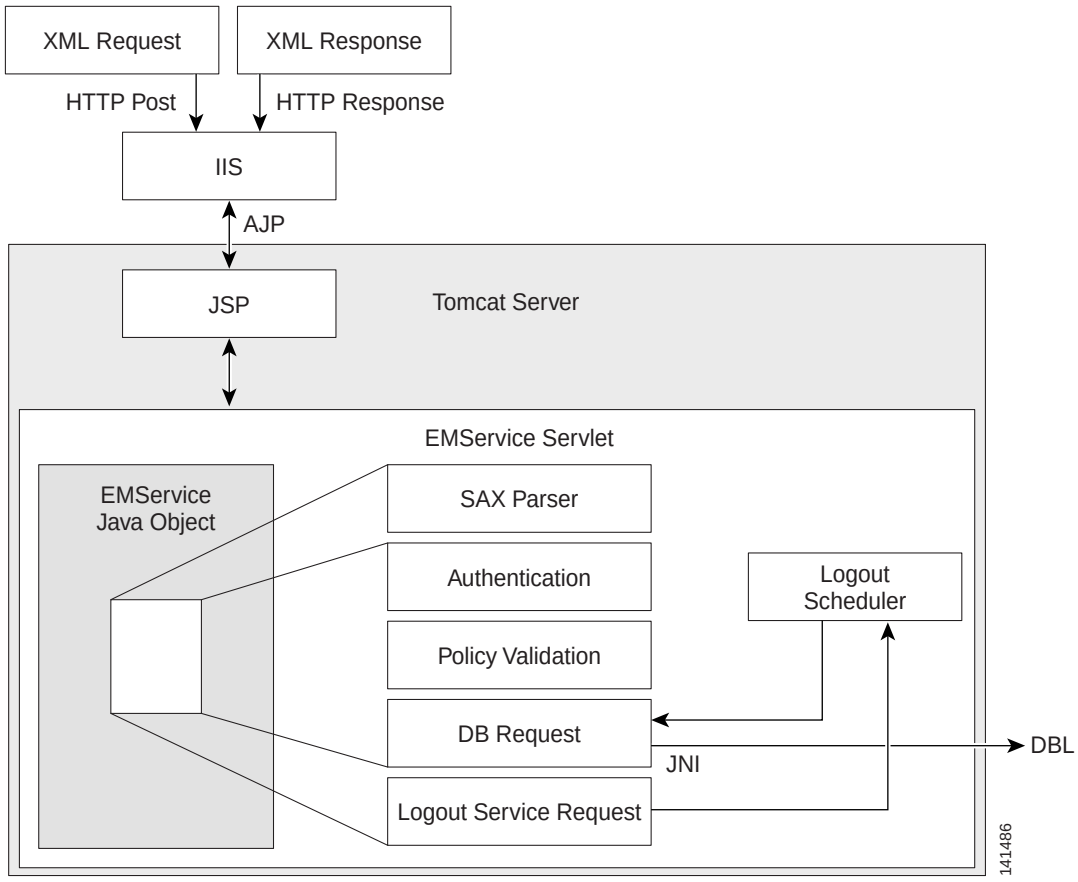
Only a single user can log in at a time on a particular device. Subsequent attempts by users to log in on a device before the previous user has logged out will fail. You also cannot log out of a device to which no user is currently logged in. These conditions generate error messages.

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 3-13

Figure 3-2 illustrates more details of the operation of the EMService module.

Figure 3-2 Cisco Unified CallManager 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 Unified CallManager Extension Mobility service allows authentication by proxy. That is, a user with Cisco Unified CallManager 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 Unified CallManager 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 CallManager Administration, User Configuration.

Pre-Conditions

The EMService Java Object's 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 Unified CallManager 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.

Logout Device Profile

Cisco Unified CallManager Extension Mobility requires a Logout Device Profile for each configured device. Cisco Unified CallManager 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's configuration. You cannot associate it with a user.
- User device profile—It is generated by an administrator and 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 Unified CallManager 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 Unified CallManager Extension Mobility service represents an IP phone service that allows a user to enter a userID and PIN at the phone itself and log into the phone. The architecture and implementation of 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 Unified CallManager 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.

This section includes the following topics:

New and Changed Information

This section describes new features and or changes that are pertinent to the Cisco Unified CallManager Extension Mobility Service in Cisco Unified CallManager Release 5.0 for API developers.

Dial Plan on SIP Phones

SIP phones will have dial plan information pushed to them via a configuration file. This dial plan exists as an unordered list of dial maps. One dial plan corresponds to each Calling Search Space (CSS).

In the case of Extension Mobility, the effective list of CSSs for a phone combines “Device CSS” and “Line CSS.” Device CSS is tied to the physical attributes of the phone like location, region, and so on, and does not change across the login and logout of users. Line CSS is tied to the lines and changes when users login or logout.

The database layer (DBL) generates the appropriate list of dialmaps when login/logout operations are performed. Because each CSS has a dial map that is linked with it, the DBL module extracts dial map information for each line in the device profile, appends this information to the dial map corresponding to the device CSS, and the resulting dial map list is pushed to the configuration file.

Survivable Remote Site Telephony (SRST) Mode

In SRST mode, the extension mobility behavior remains the same as in previous releases. Users cannot perform login and logout operations when a phone is in SRST mode. Call processing proceeds as normal, given that the SRST gateway is configured appropriately.

Credentials for SIP Digest Authentication

Credentials for SIP digest authentication are configured in the “user directory” window. All the device profiles that are associated with that user must use this set of credentials. For phones that read this information from the configuration file, DBL populates the configuration file with the appropriate credential set.

For phones that do not support digest credentials in the configuration file, digest authentication can break. For extension mobility, the SIP digest credentials change when a user logs in and out. The user must enter the credentials when placing a call.

EMApp Now a Separate Deployable Service

The Extension Mobility Application module (EMApp) remained as a servlet that was automatically started whenever the Cisco Tomcat service was running until the previous release of Cisco CallManager. This service acted as an interface between the phone and the EMService module, the component that updates the database to perform the actual login and logout.

In this release, EMApp is now a separate deployable service. This change occurred to address the serviceability assumption that each service has only one servlet. The administrator must activate/deploy the service before performing start/stop operations from the control center. The upgrade of Cisco Unified CallManager from Windows to Linux takes care of automatically activating this service.

EMApp depends on EMService and is automatically activated when EMService is activated.

The following comprise the necessary changes in the files to implement this feature:

- TypeService.csv : ccm_sw\Projects\DBL\DBLInstallCSV—This file updates the database with the correct values. A new entry for “Cisco Extension Mobility Application” is added with the service enum value of 41.
- ActiveConf.xml : ccm_sw\Common\XML\ServiceManager—This file is used by service manager and serviceability. A new entry for “Cisco Extension Mobility Application” is added and made a dependent service of EMService.
- ActiveW3C.xsd : ccm_sw\Common\XML\ServiceManager—This file is changed to include “Cisco Extension Mobility Application” in the schema.
- cm-em.spec : ccm_sw\Projects\CMAppServices\rpm—This file executes during rpm installation. Instead of webapps directory, emapp.war is copied to the appropriate location.
- ServicesConf.xml : ccm_sw\Common\XML\ServiceManager—Because EMApp has become a deployable service, the entry is removed from this file.
- ServicesW3C.xsd : ccm_sw\Common\XML\ServiceManager—EMApp entry is also removed from the schema.

Note The autologout function of Extension Mobility does not work when the user is logged in/out of EM via the AXL interface. The AXL interface is not intended to be used as a real-time API.

Failover Support

This change addresses the nonavailability of EM when a failure occurs in a node in a cluster. This restriction exists primarily because the Services button on the phone is configured to Unified CM CIP in only one node and, if that node is down, the Services button does not work.

This solution takes advantage of the fact that multiple IP addresses can be configured for a DNS entry. A new DNS entry is created with all the IP addresses of the nodes in a cluster. This new name is not related to any of the Cisco Unified CallManagers. When a failure occurs in a node, the phone contacts Unified CM CIP on the next node to get the list of services. When a user selects a service from this list, the phone will contact EM on that node.

You must make the following configuration changes to provide this support:

Step 1 Create a new hostname with more than one IP address in the DNS entry.

For example, CLUSTER with the IP address IP1, IP2, IP3 for nodes N1, N2, N3, respectively.

Unified CM CIP must run on all the nodes in the cluster. The IP phones must handle multiple IP addresses that the DNS server returns. This release enhances the way phones handle multiple IP addresses.

Step 2 Modify the URL for the IP phone service that is configured for EM:

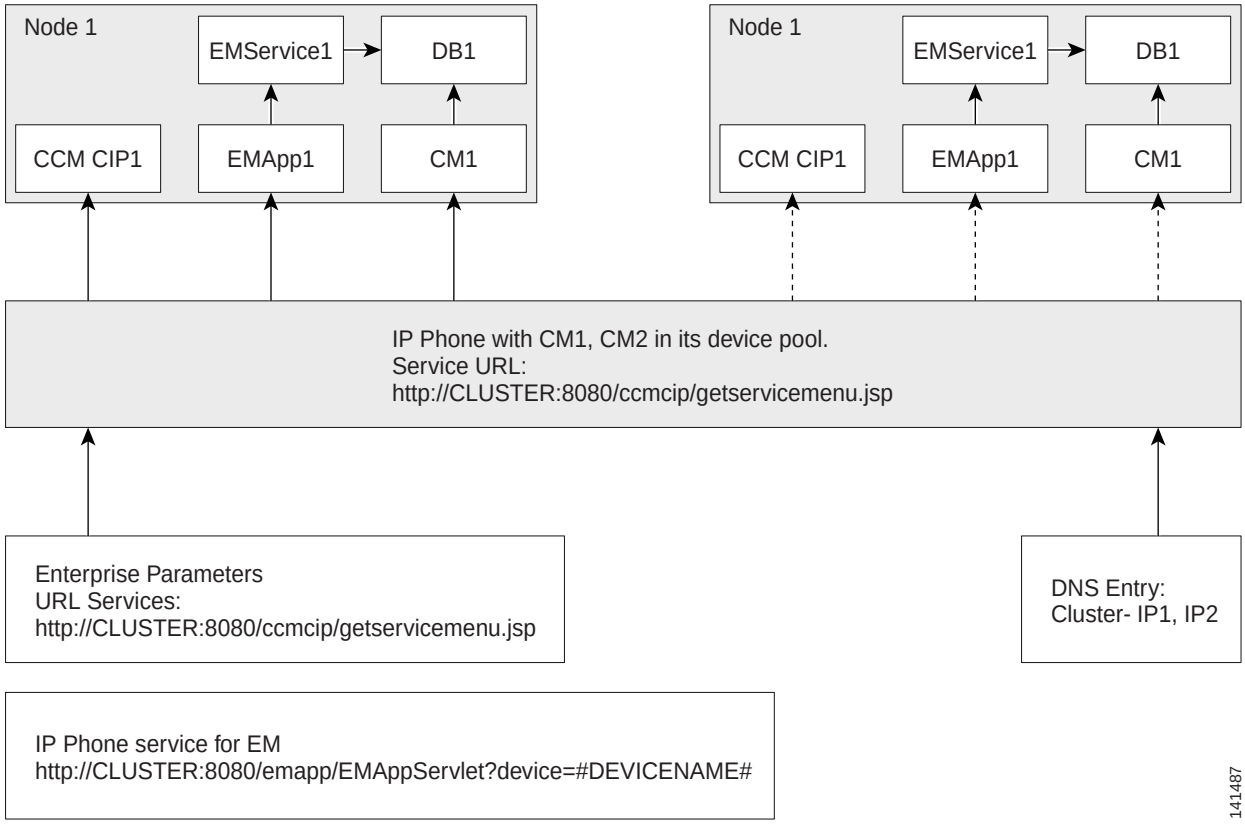
http:// CLUSTER:8080/emapp/EMAppServlet?device=#DEVICENAME#

Step 3 Change the services URL in the enterprise parameters:

http:// CLUSTER:8080/ccmcip/getservicesmenu.jsp

Figure 3-3 provides a diagram that shows the different failover support components. The dotted lines represent the connections after a failover. The shaded boxes represent required configuration changes.

Figure 3-3 Cisco Unified CallManager Extension Mobility Failover Support Components



Clearing Call Logs

The call log (missed calls, received calls, placed calls) is cleared only during manual EM login and manual EM logout. If EM logout is from ccmadmin, due to automatic logout, or from sampleloginapp, then the log is not cleared. This practice ensures privacy by preventing other users of the same phone from seeing the call logs of the previous user.

In the previous release, EM sent a "login/logout successful" message to the phone after a successful login/logout, and the phone would reset immediately. This message stays on the screen for a maximum of 10 seconds or until the phone does a reset, whichever happens first. If a reset happens sooner, the user might miss seeing this message on the screen.

In this release, lack of enough time to clear the log and send the message before the phone resets means that the call logs are cleared during both login/logout, but no message displays to show that login/logout was successful; however, in the case of failure, appropriate error messages display.

Note Call logs are cleared only during manual EM login and manual EM logout. If EM logout is from ccmadmin, due to automatic logout, or from sampleloginapp, the log is not cleared.

Last Login Stored in Database

In earlier releases, EM preserved the last login UserID of the EM user, so future logins need not enter the same user every time. This change was implemented by saving the last UserID in a local file on the hard disk. In this release, the Cisco Unified CallManager database stores this field either in the device table or in a separate table, which makes this feature more robust and secure.

The algorithm follows:

- When EMApp gets a EM login request, it sends an XML request to EMService to query the current device status.
- EMService responds to this request by returning the device status and the last logged in user for the device, if there is one.
- When there is successful EM login, EMApp sends the last login user id in the login request to EMService so that this field can be updated in the database.

Enhanced Service URL Generation

In the previous release, the services URL was queried based on the activation of the node and not the service. Node activation does not mean that the node is up and running.

In this release, EM queries the service activation table to build the list of EM services that are activated in the cluster. EM also listens to change notification whenever a new node is activated or deactivated.

When a new EM login request is sent to EMApp, it looks at the list of active EM services in the list of service URLs and sends an HTTP request to the first one. Because activation of EM services does not necessarily mean that the EMService module is actually running, the HTTP request might not reach the EMService module, and an HTTP timeout results. In this case, EMApp sends the login request to the next URL in the list of service URLs.

Localization Changes

A recent field addition to the Cisco Unified CallManager table “typeUserLocale” called “alternativeScript” contains the name of an alternative script that can be utilized to allow an endpoint to use a localized file that contains the same language but with a different written script. This change was implemented for Japanese with “kanji” as the alternative to “katakana.” Some devices can display only kanji script while some can display only katakana. Because applications can send text information to the phone, they need to know if the phone can support just the default language, or both the default languange and the alternate script. If the phone can support an alternate script, applications need to know what the alternate script is.

EM can get this information for each device model from the database or from an HTTP request. You need to make sure that the HTTP request contains this information. Based on this information, EM will send appropriate strings to the phone.

The “ProductSupportsFeature” table includs a field “useAlternateScript.” This table includes entries based on the phone model. If this field is set to 1, the “TypeUserLocale” table contains the actual script that must be used as the alternate script. Because the tables are quite small, EM could load this information and keep it in its cache or look for this information and the phone model in the HTTP request and then decide which to use.

Enhanced Error Codes

In this release the extension mobilitty feature presents the user with enhanced error codes to indicate the nature of the problem. Prior to this release, only a few error codes were displayed on the phone user interface with a description and, for others, only error codes were given. This arrangement lead to difficulties in interpreting the nature of the problem. Overlapping error codes existed between the extension mobility application and the extension mobility service. In this release, separate error codes for extension mobility application and service exist with different ranges specified. EM service error codes range from 0-199 while EM application error codes begin at 200 and range upward. These changes minimize the learning curve for the user with respect to the older error codes.

For detailed information on the new EM error codes, see the "Application and Service Error Codes" section on page 3-18.

Change Notification Enhancements

The extension mobility feature now provides change notification capability to EMApp, which previously did not have change notification for the service parameters. It also provides the capability to listen for change notifications with respect to activation and deactivation of extension mobility service on any node.

EMApp listens for Service Parameter Change notifications and EM Service activation/deactivation on any node within the cluster. This release adds a new EMActivationListener to the Unified CM DBUtil Package. Unified CM Database exposes methods to add and remove the EMActivationListener objects. Finally, the actual notification part is appended to the listener notification mechanism of Unified CM Database.

The notification mechanism for EMActivationListener differs from the ServiceActivationListener. ServiceActivationListener is used primarily for listening for events regarding local service activation or deactivation. It does not propagate events regarding service activation on any other node. EMActivationListener notification mechanism actually sends all events regarding EMActivation on any node to the interested observer (EMApp).

EMApp listens for the activation/deactivation details of the EMService module and maintains a memory map on the activation status of all the EMService modules on different nodes. The system consults this memory map before actually sending a request to that particular EMService module on the node.

Performance Counter Enhancements

This change gives a more granular mechanism to measure extension mobility performance. This release adds four new counters:

- EMAPP_ATTEMPTED_LOGIN_LOGOUT—Measures the total number of attempted login and logout requests. This counter includes both successful and failed requests. This counter does not measure the queries that are posted. Thus this count represents the total login and logout requests that were attempted on this node.
- EMAPP_NO_OF_SUCESS_LOGIN—Captures the successful login requests.
- EMAPP_NO_OF_SUCCESS_LOGOUT—Captures the successful logout requests (only coming through requests). This counter does not include autologouts.
- EMAPP_NO_OF_THROTTLED_REQUESTS—Measures all throttled requests (throttling applies only to login/logout requests).

You can use these counters for calculating the success and failure rate of extension mobility logins:

Total Number of Successful Requests =
EMAPP_NO_OF_SUCESS_LOGIN + EMAPP_NO_OF_SUCCESS_LOGOUT

Total Request Failures (Only Login/Logout) =
EMAPP_ATTEMPTED_LOGIN_LOGOUT - Total Number of Successful Requests

Request Failures Other than Throttling =
Total Request Failures - EMAPP_NO_OF_THROTTLED_REQUESTS

EMPerfmon.xml (/Projects/serviceability/pi/xml/) contains all the perfmon object definitions for extension mobility. You can view the perfmon counters by using the RTMT tool.

SIP Phones

No basic design changes occurred from either the EMApp or EMService perspective as far as SIP phones are concerned. However, successful login through EM on SIP phones depends on several factors:

- SIP phone support for XSI interfaces. SIP phones must support the following set of URI's to provide the basic functionality of EM through EMApp:
  - CiscoIPPhoneText
  - CiscoIPPhoneInput
  - CiscoIPPhoneExecute
  - CiscoIPPhoneMenu
  - Key:Services
  - SoftKey:Exit
  - Init:Services
  - Init:CallHistory
- TFTP should be able to generate appropriate configuration files as described in the “TFTP Process Overview for Cisco SIP IP Phones” section in the “Cisco TFTP” chapter in the *Cisco Unified CallManager System Guide for Release 5.0*.

Note Legacy phones with SIP loads cannot support EM due to limitations with the XML object interface; however, SIP loads on Cisco Unified IP 7961 and 7970 phones are expected to support EM.

Configuration

The Extension Mobility service application accompanies Cisco Unified CallManager. As such, all necessary Cisco Unified CallManager Extension Mobility service API components get installed with the standard Cisco Unified CallManager installation.

To use the Cisco Unified CallManager 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 Unified CallManager Extension Mobility service:

- Activate the service
- Create EM 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 EM and configure the default device profile on the target device. (You must enable EM on a device-by-device basis.)
- Subscribe to EM IP phone service on the target device and the device profile.
- Assign a logout device profile to a target device.
- Configure the system parameters (defaults are used 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 CallManager Administration Guide* or *Cisco Unified CallManager 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 set by the Message DTDs that are described in the "Message Document Type Definitions" section on page 3-14.

The default URL for login and logout requests and system queries is:

http://<server>/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 legal Cisco Extension Mobility messages, see the "Message Examples" section on page 3-15.

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 3-15.

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 3-2 lists each example type, a description of the example, and a reference to the example.

Table 3-2 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

Table 3-2 Message Examples (continued)

Message Type	Description	Reference
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 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 3-3 shows the new error codes the Cisco Unified CallManager Extension Mobility Application module returns as well as the error code numbers for the previous release, and describes what each error code means.

Table 3-3 Cisco Unified CallManager 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().

Table 3-3 Cisco Unified CallManager Extension Mobility Application Error Codes (continued)

New Error Code	Previous Error Code	Message—Description
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 3-4 shows the current (new) error codes that the Cisco Unified CallManager 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 3-4 Cisco Unified CallManager 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.
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 extension mobility service received an error while trying to communicate with the Cisco Unified CallManager database.
7	7	LOGOUT_REQUEST_ERROR—Not used.

Table 3-4 Cisco Unified CallManager Extension Mobility Service Error Codes (continued)

New Error Code	Previous Error Code	Message—Description
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
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 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.

Table 3-4 Cisco Unified CallManager Extension Mobility Service Error Codes (continued)

New Error Code	Previous Error Code	Message—Description
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.

WebDialer API Programming

This chapter describes the Simple Object Access Protocol (SOAP) and HTML over HTTP (and HTTPS) interfaces used to develop customized directory search applications for Cisco Unified CallManager WebDialer Version 1.2, and contains the following sections:

Definitions

Cisco WebDialer Server	A server that hosts the Cisco Unified CallManager WebDialer application
Cisco WebDialer Application	The software that is installed on a Cisco Unified CallManager 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 on the telephone number of the person they are trying to call. The Cisco Unified CallManager WebDialer application consists of two Java servlets, the WebDialer servlet and the Redirector servlet.
WebDialer Servlet	A Java servlet that allows Cisco Unified CallManager users in a specific cluster to make and end calls, as well as access their phone and line configuration.
Redirector Servlet	A Java servlet that finds the Cisco Unified CallManager cluster for a request made by a Cisco WebDialer user. It redirects that request to the specific Cisco WebDialer server located in that user's Cisco Unified CallManager cluster.

Overview

Cisco Unified CallManager WebDialer, which is installed on a Cisco Unified CallManager server and used in conjunction with Cisco Unified CallManager, allows Cisco Unified IP Phone users to make calls from web and desktop applications. For example, Cisco Unified CallManager WebDialer uses hyperlinked telephone numbers in a company directory to allow users to make calls from a web page by clicking on the telephone number of the person they are trying to call.

The two main components of Cisco Unified CallManager WebDialer are the Webdialer Servlet and the Redirector Servlet.

Webdialer Servlet

The WebDialer servlet is a Java servlet that allows Cisco Unified CallManager users in a specific cluster to make and end calls, as well as to access their phone and line configuration.

Cisco WebDialer applications interact with the WebDialer servlet through two interfaces:

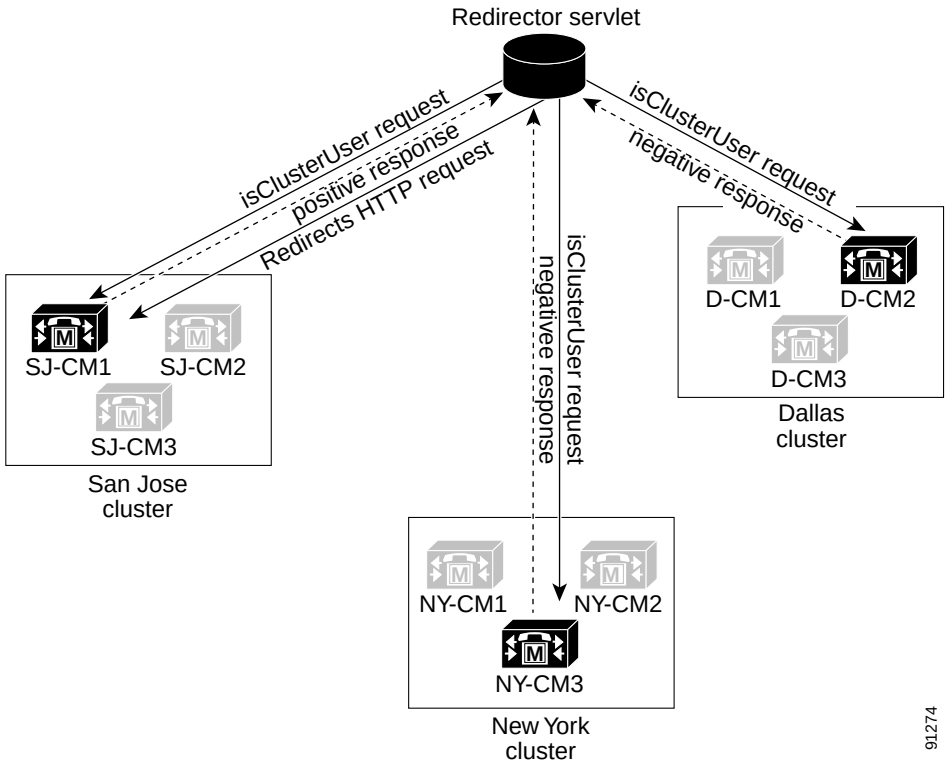
- The SOAP over HTTP interface—This interface that is based on the Simple Object Access Protocol (SOAP) gets used to develop desktop applications such as Microsoft Outlook Add-in and SameTime Client Plug-in. Developers can use the isClusterUserSoap interface to design multicluster applications that require functionality similar to a Redirector servlet.
- HTML over HTTP interface—This interface that is based on the HTTP protocol gets used to develop web-based applications such as the Cisco Unified CallManager 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 CallManager cluster for a request made by a Cisco WebDialer user. It redirects that request to the specific Cisco WebDialer server that is located in that user's Cisco Unified CallManager cluster. Availability of the Redirector servlet occurs only for multicluster applications and only for applications that are developed by using HTML over HTTP or secure HTTP (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 Unified CallManager WebDialer 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 CallManager servers with Webdialer servlets that have been configured for Cisco Unified CallManager servers SJ-CM1, D-CM2, and NY-CM3.

The system administrator configures the Webdialer servlets on any of the Cisco Unified CallManager servers by entering the IP address of that specific Cisco Unified CallManager server in the *wdservers* service parameter.

For information on configuring Webdialer and Redirector servlets, refer to the *Cisco WebDialer* chapter in the *Cisco Unified CallManager Features and Services Guide, Release* 5.0.

When a user who is located in San Jose clicks on a telephone number in the corporate directory search page that is enabled by Cisco Unified CallManager WebDialer, the following actions happen:

1. The Cisco Unified CallManager server sends an initial *makeCall* HTTP request to the Redirector servlet.
2. If this request is received for the first time, the Redirector servlet reads the Cisco Unified CallManager WebDialer server cookie and finds it empty.

   For a repeat request, the Redirector servlet reads the IP address of the Cisco Unified CallManager WebDialer server that previously serviced the client and sends a *isClusterUser* HTTP request only to that server.
3. The Redirector servlet sends back a response asking for information, which results in the authentication dialog box opening for the user.
4. The user enters the Cisco Unified CallManager 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* HTTP request to each Cisco Unified CallManager WebDialer server that has been configured by the system administrator.

   Figure 4-1 illustrates how this request is sent to the Webdialer servlets that have been configured for SJ-CM1, D-CM2, and NY-CM3. Depending on the geographical location of the calling party, the WebDialer servlet from the cluster representing that location responds positively to the Redirector servlet. The remaining Webdialer servlets that were contacted return a negative response. The WebDialer 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's browser for future use.

Changes in Release 5.0

The following changes to the Cisco Unified CallManager WebDialer were included in Cisco Unified CallManager Release 5.0.

- Logging mechanism modified to apply changes to both WebDialer and Redirector
- Redirector fully functional with HTTPS. The previous support for HTTPS was only partial.
- WebDialer should subscribe for Dial Rule Change Notification.
- Provided migration of Dial Rule data during an upgrade from Windows to Linux.
- WebDialer support is provided for enhancements that were made in CTI and JTAPI.
- Provided SIP support in WebDialer
- Increased thread count to 3.

These enhancements are explained in detail in the sections that follow.

Logging Mechanism Changes

In the previous release the changes made to the logging were applied only to the Webdialer servlet. In this release the logging mechanism has been modified so that the changes are applicable to both the WebDialer and Redirector servlet.

The serviceability mechanism provided for applying the changes in one logger to another logger has been used. The fappender.jar is now included in the WebDialer class path to take care of this dependency.

Redirector Over HTTPS

Redirector support for HTTPS was only partial until the previous release. Requests to the WebDialer server were using HTTP as the protocol. Redirector now supports HTTPS completely.

Dial Rule Change Notification

In previous releases WebDialer was not subscribed to Change Notification from the Application dial rule table. The WebDialer service had to be restarted every time an add/update/delete was made to the dial rules. Starting with this release WebDialer subscribes to Change Notification from the Application dial rule table.

WebDialer Support for Enhancements in CTI and JTAPI

The following enhancements in CTI and JTAPI are supported in WebDialer:

- WebDialer includes the necessary changes to accommodate the modified interfaces for QoS and SRTP.
- WebDialer supports secure connections to CTI (TLS connection). For this feature, webdialer will be using the security API provided by JTAPI. Please refer to the *Cisco Unified JTAPI Developers Guide* for the JTAPI API. There is also a new Application User, "WDSysUser", which is used by WebDialer for obtaining the CTI connection.

  The following configuration must be completed before WebDialer can be configured to open a CTI connection in secure mode.

Step 1 Activate the "Cisco CTL Provider " service from the Cisco Unified CallManager Service administration page.

Step 2 Activate the "Cisco Certificate Authority Proxy Function" Service.

Step 3 Download the "Cisco CTL Client" from the Application plugin and install it on any machine.

Step 4 Run the CTL Client, choose the option to "enable Cluster Security" and follow the instructions that are displayed. This requires USB E-tokens.

Step 5 To verify that cluster security is enabled, go to the Cisco Unified CallManager administration page and look at [System-> Enterprise Parameter configuration]. Look at the Security Parameters; the cluster security should be set to 1.

Step 6 In the Cisco Unified CallManager administration page, from the "UserManagement" drop-down menu select the "Application User CAPF Profile" option.

Step 7 Click "Add new" InstanceID.

Step 8 In the CAPF Profile configuration wndow, setup 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 is past, then 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.

New Service Parameters

WebDialer has two new service parameters for CTI connection security. These are node-specific parameters.

- **CTI Manager Connection Security Flag**—This required service parameter indicates whether security for the Cisco Unified CallManager WebDialer service CTI Manager connection is enabled or disabled.

  If enabled (true), Cisco Webdialer will open a secure connection to CTI Manager using the Application CAPF profile configured for the instance ID (as configured in CTI Manager Connection Instance ID service parameter) for Application user WDSysUser. The default value is 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 Unified CallManager WebDialer 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', then delete the current certificates, if any.
  4. If the Certificate Operation is not 'Install/Upgrade' or 'Delete', and there is a current certificate, use this certificate.
  5. If there is no certificate present, request one 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, then set the following 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, then throw initFailedException. This can be seen in the WebDialer traces.

Install Changes

WebDialer 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
```

Partition Support

WebDialer supports the partition information that is now provided by JTAPI. The WebDialer preferences page now includes information about the partition along with the line.

Migration of Dial Rules Data

The directory exports data into a CSV file. The CSV file points to a file in the same location, which has the dial rules in binary format as they are stored in the directory. The post-install script for WebDialer parses this CSV file, finds the file that has the encoded dial rules, decodes them, and populates the database with the dial rules.

SIP Support in WebDialer

WebDialer 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 WebDialer. The APIs provided by JTAPI are used to distinguish between these two kinds of phones and hide the unsupported phones from the user on the WebDialer preferences page.

Increase in Thread Count to 3

In previous releases, WebDialer supported only two concurrent threads. The performance testing for WebDialer shows better results (in terms of successful call rate) if the thread count is increased to 3. To increase performance, this change is being made for WebDialer in this release.

Call Flows

The call flows in this section describe the flow of events for client and browser-based applications that use Cisco Unified CallManager WebDialer, and should help you design customized applications for Cisco Unified CallManager WebDialer.

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 WebDialer 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:

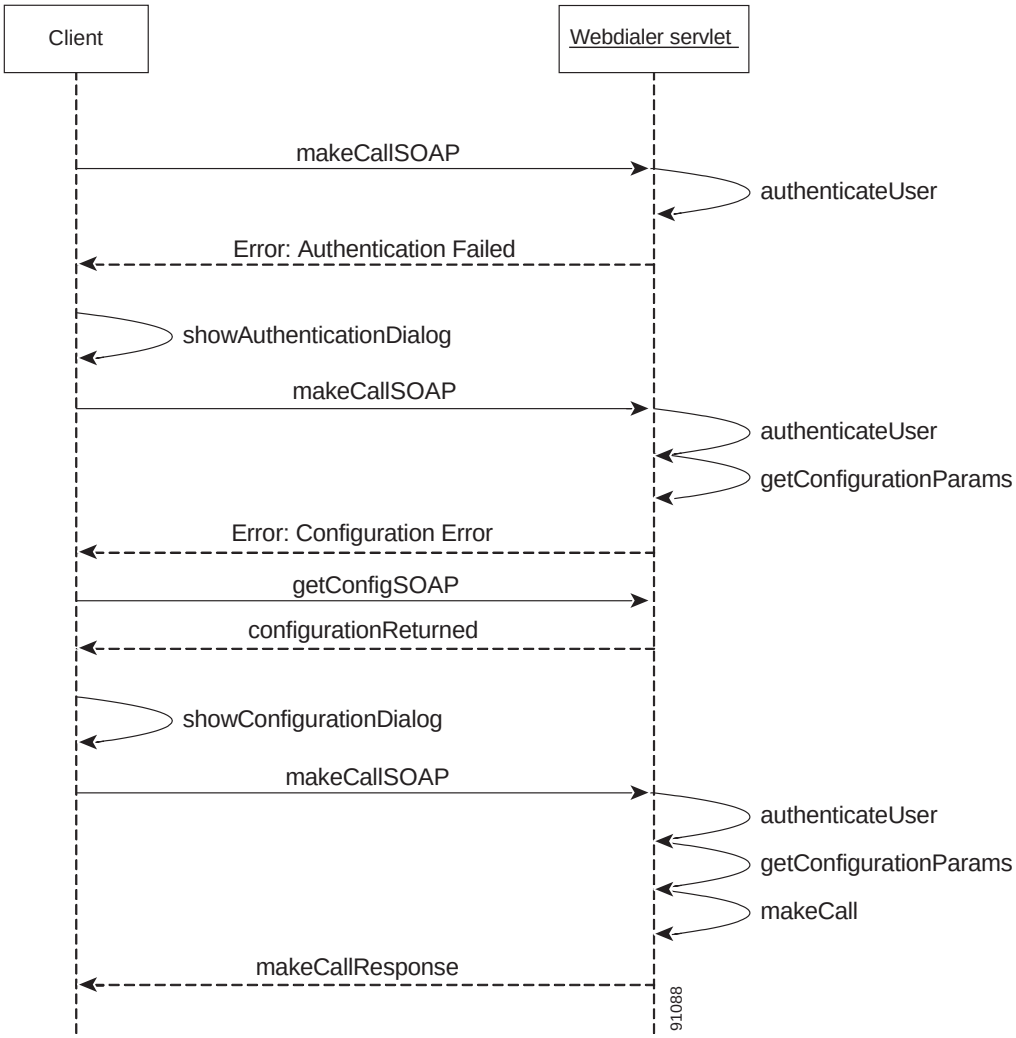
1. The client sends a makeCallSoap request to the configured WebDialer servlet.
2. The WebDialer servlet attempts to authenticate the user. Figure 4-2 shows an authentication failure because the authentication information is incomplete or does not exist.
3. The WebDialer servlet sends an authentication failure response to the client application.
4. The client application displays a dialog box on the computer screen of the user asking for the user ID and password. The user enters this information and clicks the **submit** button. The user ID and password is now stored for future invocations of the application.
5. The application sends a repeat SOAP request to the WebDialer servlet. The request contains credential information on the user.
6. The WebDialer servlet authenticates the user.
7. The WebDialer servlet reads any missing configuration information in the request.
8. The WebDialer servlet returns a configuration error message to the client application.
9. The client application sends a getConfigSoap request to the WebDialer servlet.
10. The WebDialer servlet responds with the user's configuration information stored in the directory.
11. The client application displays a configuration dialog box on the user's computer screen asking the user to select or update the configuration. The user enters the information and clicks the **submit** button. The user's configuration information is now stored for future invocations of the application.
12. The client resends the makeCallSoap request to the WebDialer servlet. This request contains the user's configuration information.

13. The WebDialer servlet authenticates the user and dials the telephone number using the information contained in the makeCallSoap request. 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 made by the user.

Figure 4-2 Cisco Unified CallManager WebDialer Call Flow for a Client-based Application

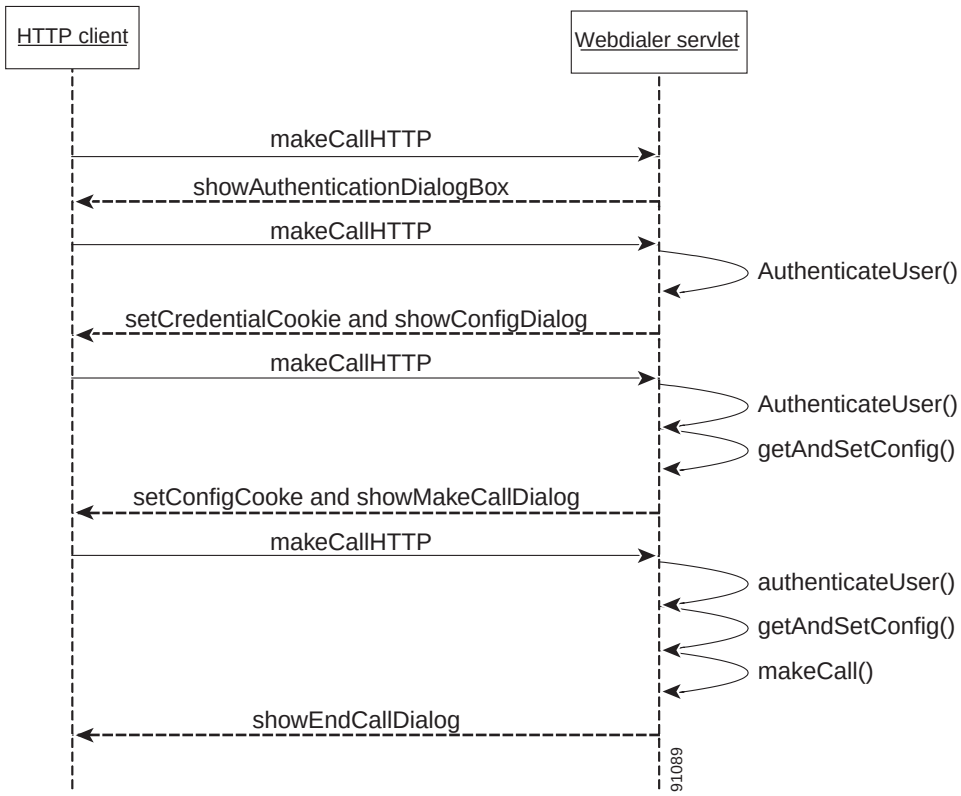


Browser-based Application Call Flow

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 CallManager 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 Unified CallManager WebDialer Call Flow for a Browser-based Application



If the user makes a call for the first time:

1. The client sends a makeCall HTTP request to the configured WebDialer servlet. The query string contains the number to be called.
2. The WebDialer servlet authenticates the user. Authentication fails because the authentication information is incomplete or does not exist.

Note Authentication is successful if the user's credentials are sent with the request, and the call flow goes directly to number seven.

3. The WebDialer 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 makeCallHTTP request containing the user's credentials to the WebDialer servlet.
6. The WebDialer servlet authenticates the user.
7. The WebDialer servlet reads the configuration information in the cookie which is sent with the request.
8. Assuming that the request is made for the first time, the servlet sends a response containing a cookie to the client's browser. The cookie containing the client's credentials is stored on the client's browser. The client's credentials are 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's browser sends a makeCall HTTP request to the WebDialer servlet. The request contains a cookie with the credential and configuration information in parameter form.

11. The WebDialer servlet uses the credentials to authenticate the user and saves the configuration information in its memory.
12. The WebDialer servlet sends a makeCall confirmation dialog to the client's browser with the configuration information stored in a cookie. The cookie is stored on the client's browser for future invocations.
13. The Make Call dialog box appears on the user's computer screen. The user clicks the **Dial** button which sends another makeCall HTTP request to the WebDialer servlet.
14. The WebDialer servlet authenticates the user 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's computer screen and stays there for the amount of time configured in the service parameters.

For all subsequent requests, the call flow starts at number 12 and ends at number 15.

Interfaces

Cisco Unified CallManager WebDialer applications interact with the WebDialer servlet through two interfaces:

- **SOAP over HTTP** interface— This interface is based on the Simple Object Access Protocol (SOAP) and 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 multi-cluster applications that require functionality similar to a Redirector servlet.
- **HTML over HTTP** interface— This interface is based on the HTTP protocol and is used to develop web-based applications such as the Cisco Unified CallManager directory search page (directory.asp). Developers using this interface can use the Redirector servlet for designing multi-cluster applications.

Note The following files must be run to properly set the ENV variable and Java classes:
installWDService.bat
installWDSOAP.bat

SOAP over HTTP Interface

To access the SOAP interfaces for Cisco Unified CallManager WebDialer, use the Cisco Unified CallManager WebDialer Web Service Definition Language (WSDL) in the "WebDialer WSDL" section on page 4-17.

makeCallSoap

The makeCallSoap interface is accessed by initiating a SOAP request to the URL http://CCM-IP:8080/webdialer/services/WebdialerSoapService where CCM-IP is the IP address of the Cisco Unified CallManager server on which WebDialer 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.	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 WebDialer* chapter in the *Cisco Unified CallManager Features and Services Guide, Release 5.0*.	Refer to the credential data type in the "WebDialer WSDL" section on page 4-17.	None	None
Profile	Mandatory	The profile that is used to make a call. An example of a typical profile is a calling device such as an IP phone or line.	Refer to the profile data type in the "WebDialer WSDL" section on page 4-17.	None	None

Results

Refer to the "WebDialer 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	
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, there are missing parameters 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, there are missing parameters in the request.	

Error Code	Name	Type	Description	Action by application
6	responseCode	Integer	Service is temporarily unavailable.	Displays the appropriate error dialog with an option to try again.
	responseDescription	String	Service is 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	

endCallSoap

The endCallSoap interface is accessed by initiating a SOAP request to the URL http://CCM-IP:8080/webdialer/services/WebdialerSoapService where CCM_IP is the IP address of the Cisco Unified CallManager server on which 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 WebDialer* chapter in the *Cisco Unified CallManager Features and Services Guide, Release 5.0*.	Refer to the credential data type in "WebDialer WSDL" section on page 4-17.	None	None
Profile	Mandatory	The profile that is used to make a call. An example of a typical profile is a calling device such as an IP phone or line.	Refer to the profile data type in the "WebDialer WSDL" section on page 4-17.	None	None

Refer to the "WebDialer 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, there are missing parameters 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, there are missing parameters in the request.	
6	responseCode	Integer	Service is temporarily unavailable.	Displays the appropriate error dialog with an option to try again.
	responseDescription	String	Service is 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

The getProfileSoap interface is used by plug-in based clients and accessed by initiating a SOAP request to the URL http://CCM-IP:8080/webdialer/services/WebdialerSoapService where CCM-IP is the IP address of the Cisco Unified CallManager server on which WebDialer 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 WebDialer* chapter in *Cisco Unified CallManager Features and Services Guide, Release 5.0*.	Refer to the credential data type in the "WebDialer 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 "WebDialer 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 associated with the user. Refer to the Cisco Unified CallManager WebDialer 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 is configured for the user.	Displays an appropriate error message.
	responseDescription	String	No device is 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 is temporarily unavailable.	Displays the appropriate error dialog with an option to try again.
	responseDescription	String	Service is temporarily unavailable.	
9	responseCode	Integer	Service is overloaded.	Displays the appropriate error dialog with an option to try again.
	responseDescription	String	Service is overloaded.	

isClusterUserSoap

The isClusterUserSoap interface is accessed by initiating a SOAP request to the URL http://CCM-IP:8080/webdialer/services/WebdialerSoapService where CCM-IP is the IP address of the Cisco Unified CallManager server on which WebDialer is configured.

This SOAP interface is used for multi-cluster applications that require functionality, similar to a Redirector servlet, for redirecting calls to the various locations where Cisco Unified CallManager WebDialer is installed on a network. The application uses this interface to locate and verify the Cisco Unified CallManager WebDialer servicing the user, followed by makeCall, endCall or getProfile requests to that WebDialer.

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 "WebDialer WSDL" section on page 4-17 for return values and their data type.

Name	Type	Description
result	Boolean	The result is true if the user is present in the directory of the cluster. The result is false if the user is not present in the directory.

HTML over HTTP Interfaces

This section describes the HTML over HTTP interfaces.

makeCall

The makeCall interface is used in customized directory search applications. It is also used by the Cisco Unified CallManager directory search page (directory.asp). The makeCall interface is accessed by initiating an HTTP request to the URL http://<ipaddress>/webdialer/Webdialer. In this URL, ipaddress is the IP address of the Cisco Unified CallManager server for which WebDialer is configured.

This interface is used by browser-based applications in which the browser accepts cookies. 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 is converted to a regular telephone number by applying the application dial rules. Refer to the *Cisco WebDialer* chapter in the *Cisco Unified CallManager Features and Services Guide, Release 5.0*.	String	None	None

Name	Description
result	Cisco Unified CallManager WebDialer displays the appropriate dialog and its applicable success or error message. It displays an authentication dialog if there is no active session.

makeCallProxy

The makeCallProxy interface is accessed by initiating an HTTP request to the URL http://ipaddress/webdialer/Webdialer?cmd=doMakeCallProxy. This interface is used by browser-based applications in which the browser accepts cookies. If the cookies are disabled in a browser, the user profile exists for only the length of the session.

The makeCallProxy interface can be used by applications such as a personal address book, defined in theUnified CMUser pages at http://cmserver/CMUser. The credential of the application is used, as a proxy, to make calls on behalf of users. Since these users have authenticated themselves before accessing theUnified CMUser page, they are not 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.

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 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 pages. After this login, other pages do not require the user to log in again. For web page applications that are not integrated, appid is the same as userid.	String	None	None
pwd	Mandatory	The password of the appid.	String	None	None
destination	Mandatory	The number to be called. This number is converted to an E.164 number by the dial plan service.	String	None	None

Name	Description
result	Cisco Unified CallManager WebDialer displays the appropriate dialog and its applicable success or error message.

WebDialer WSDL

The Web Service Definition Language (WSDL) for Cisco Unified CallManager WebDialer mentioned below is based on the WSDL specification. This WSDL for Cisco Unified CallManager WebDialer is also available on the Cisco Unified CallManager WebDialer server installation at:

http://CCM-IP:8080/webdialer/services/WebdialerSoapService?wsdl

Use this specific WSDL and the interfaces mentioned in this document to develop customized applications for Cisco Unified CallManager WebDialer. For a list of references on Cisco Unified CallManager, SOAP, and WSDL, refer to the "Related Documentation" section in the Preface.

```
<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="description" 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="responseCode" type="xsd:int"/>
                    <xsd:element name="description" nillable="true" type="xsd:string"/>
                    <xsd:element name="deviceInfoList" nillable="true"
type="tns:ArrayOfWDDeviceInfo"/>
                </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="Result" 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="Result" 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="Result" 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="Result" 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=
"http://WebDialer_ip_address:8080/webdialer/services/WebdialerSoapService"/>
        </port>
    </service>
</wsdl:definitions>
```

Sample JavaScript

This JavaScript is a sample script that is used to enable Cisco Unified CallManager WebDialer from a directory search page.

Single Cluster Applications

This script is used 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 = 'http://<%=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

This script is used 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= 'http://<%=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>
```

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

W