



# AXL Programming Guide for Cisco Unified CallManager 4.2(3)

---

To access all AXL SOAP API downloads and AXL requests and responses that are found in this document, refer to the following URL:

[http://www.cisco.com/cgi-bin/dev\\_support/access\\_level/product\\_support](http://www.cisco.com/cgi-bin/dev_support/access_level/product_support)

This document contains the following sections:

- [Introduction, page 2](#)
- [Target Audience for this Guide, page 2](#)
- [New and Changed Information, page 3](#)
- [AXL API, page 3](#)
- [Examples of AXL Requests, page 5](#)
- [Throttling of Requests, page 10](#)
- [The AXL Schema Documentation, page 11](#)
- [Example XML Structure, page 12](#)
- [Authentication, page 13](#)
- [Data Encryption, page 13](#)
- [Integration Considerations and Inter-Operability, page 13](#)
- [Obtaining Documentation, page 14](#)
- [Documentation Feedback, page 15](#)
- [Cisco Product Security Overview, page 15](#)
- [Obtaining Technical Assistance, page 16](#)
- [Obtaining Additional Publications and Information, page 17](#)



---

**Corporate Headquarters:**  
**Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA**

Copyright © 2006 Cisco Systems, Inc. All rights reserved.

# 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 DLLs.

The AXL API methods, known as requests, get performed by 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 Guide

This programming guide 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 guide assumes that the developer has knowledge of a high-level programming language such as C++, Java, or an equivalent language. As developer, 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 and this programming guide must have a firm grasp of XML Schema, which was 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 gives much latitude to developers to modify the Cisco Unified CallManager system database. Because each API call impacts the system, developers must take caution when using AXL. Misuse of the API can lead to dropped calls and slower system performance. Be aware that AXL is not intended as a real-time API, but as a provisioning and configuration API.

---

# New and Changed Information

This section describes the new or changed API calls for Cisco Unified CallManager Release 4.2(2) and a new service parameter for Cisco Database Layer Monitor. No new or changed API calls exist for Cisco Unified CallManager Release 4.2(3).

## Added API Calls

No new API calls exist for Cisco Unified CallManager Release 4.2(2).

## Changed API Calls

Changes to the following API calls in Release 4.2(2) support the Hold Reversion feature:

- addDevicePool (adds revertPriority to this request)
- updateDevicePool (adds revertPriority to this request)
- addLine (adds hrDuration and hrInterval to this request)
- updateLine (adds hrDuration and hrInterval to this request)

Because these new tags are disabled by default, these changes are backward-compatible with existing user code.

## New Service Parameter

A new service parameter, "EnableAXLEncodingInfo," has been added to Cisco Unified CallManager Administration under the Cisco Database Layer Monitor service. This parameter enables the user to decide if AXL responses should contain encoding information. Encoding information is important if an AXL request has non-English characters in it.

## AXL API

Request methods represent XML structures that are passed to the AXL API server. The server receives the XML structures and executes the request. If the request completes successfully, the appropriate AXL response gets returned. All responses get named identically to the associated requests, except that the word "Response" is appended. For example, the XML response that is returned from an addPhone request gets called addPhoneResponse. See [“Examples of AXL Requests” section on page 5](#) for more information.

If an error occurs, an XML error structure gets returned wrapped inside a SOAP Fault structure (see [“AXL Error Codes” section on page 4](#)).



### Note

Changes to the Cisco Unified CallManager database via AXL that require a directory change may take several seconds to complete after AXL returns a success response for the operation. For any AXL operation that triggers a directory change, such as addPhone, updatePhone, or removePhone, check to ensure that the requested change has been made by querying the user information before relying on the directory data.

## 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 gets 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 example shows 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/CCMApi/AXL/V1/axlsoap.xsd">
        <axl:error sequence="1234">
          <code>0</code>
          <message>
            <![CDATA[
              Device not found with name SEP003094C39708.
            ]]>
          </message>
        </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 provides 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 sections describe the subelements of the <error> tag.

#### code

The <code> element represents a numerical value that is used by the user agent 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 processing the request. This can occur due to a problem on the server but can also be caused by errors in the request.
5002	Unknown Request Error—This error occurs if the user agent submits a request that is unknown to the API.
5003	Invalid Value Exception—This error occurs if an invalid value is detected in the XML request.
5004	AXL Unavailable Exception—This error occurs if the AXL service is too busy to handle the request at that time. Send the request again at a later time.
5005	Unexpected Node Exception—This error occurs if the server encounters an unexpected element. For example, if the server expects the next node to be <name>, but encounters <protocol>, this error gets returned. Malformed requests that do not adhere to the latest AXL Schema always cause these errors.

#### message

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

#### request

The <request> element allows the user agent to determine what type of request generated this error. Because this element is optional, it may not always appear.

## Examples of AXL Requests

The following examples describe how to make an AXL request and how to read back the response to the request. Each SOAP request must be sent to the web server via an HTTP POST. The endpoint URL specifies an ISAPI Extension DLL. The following list contains the only four required HTTP headers:

- POST /CCMApi/AXL/V1/soapisapi.dll

The first header specifies that this particular POST is intended for the AXL SOAP API. 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 gets returned to the client.

- Authorization: Basic <some Base64 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 gets 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 gets returned.

From the AXL side, no limit exists on the size of response or the number of records in a listXXX response. A limitation may exist in the the underlying protocols; however, the AXL API 'listUserByName' returns a maximum of 500 users.

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

```
POST /CCMApi/AXL/V1/soapisapi.dll
Host: axl.myhost.com:80
Accept: text/*
Authorization: Basic bGFycnk6Y3VybHkgYW5kIG1vZQ==
Content-type: text/xml
Content-length: 613
```

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

```
POST /CCMApi/AXL/V1/soapisapi.dll
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://gkar.cisco.com/schema/axlsoap.xsd"
sequence="1234">
      <phoneNumber>SEP222222222245</phoneNumber>
    </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 back, outputting 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 = WSASStartup (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 ocured 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://gkar.cisco.com/schema/axlsoap.xsd" sequence="1234"> \
<phoneNumber>SEP222222222245</phoneNumber> \
</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 HTTP header
strHTTPHeader = "POST /CCMApi/AXL/V1/soapisapi.dll\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 HTTP 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 iTotRead = 0;
// read BUFSIZE at a time, writing to another ostream
do {
iNumRead = recv (Socket, buff, BUFSIZE-1, 0);
buff[iNumRead] = NULL;

cout << buff;
iTotRead += iNumRead;

} while (iNumRead == BUFSIZE-1);

cout << "Read " << iTotRead << " bytes." << endl;

}
else
{
cout << "An error ocured 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,
                                                // HTTP 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

        server

        // Build the HTTP Header
        sAXLSOAPRequest = "POST /CCMApi/AXL/V1/soapisapi.dll\r\n";
        sAXLSOAPRequest += "Host: localhost:80\r\n";
        sAXLSOAPRequest += "Authorization: Basic bGFycnk6Y3VybHkgYW5kIGlvZQ==\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://gkar.cisco.com/schema/axlsoap.xsd\" ";
        sAXLRequest += "sequence=\"1234\"> <phoneNumber>SEP22222222245</phoneNumber> ";
        sAXLRequest += "</axl:getPhone> </SOAP-ENV:Body> </SOAP-ENV:Envelope>";

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

        // now add the SOAP payload to the HTTP 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;
    }
}

```

## 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. As administrator, you use the Database Layer service parameter "MaxAXLWritesPerMinute" to control this value.

AXL accommodates all requests until the "MaxAXLWritesPerMinute" value is reached. Subsequent attempts to modify the database with AXL are rejected with a response as "Not enough resources to handle request. Try again later." Every minute, AXL resets its internal counter and begins to accept AXL update requests until the limit gets reached again.

The MaxAXLWritesPerMinute parameter has a default value of 50 and maximum value of 999.

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

**Note**

Despite having this throttle mechanism the applications can cause a CPU spike on the Cisco Unified CallManager depending on the rate at which the AXL requests are sent to the CallManager .

For example let us assume that the throttle is set at 60 transactions per min. This means that the AXL interface does not accept more than 60 transactions in a one minute time window. Let an application try to send 60 transactions within the first second of the 60 seconds window. Even though the interface responds back with a 503 error message after the 60th request, the rate (that is 60 requests per second = 3600 transactions per minute) at which the application sends the requests can affect the CPU performance on the CallManager.

## The AXL Schema Documentation

The complete AXL schema (including details of all requests, responses, XML objects, data types, and so on) gets encapsulated in the following four files that are found on the Cisco Unified CallManager server: axlsoap.xsd, axl.xsd, axlmessage.xsd, and AXLEnums.xsd. The default path follows:  
**C:\CiscoWebs\API\AXL\V1.**

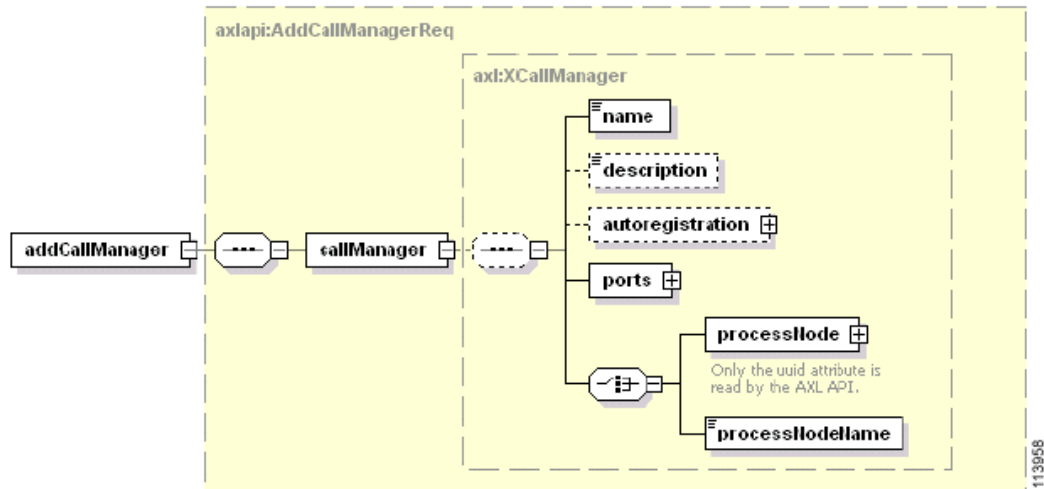
Standard XML handling IDEs and development environments can illuminate or auto-generate 'friendly' formatted documents based on the AXL schema files.

The following example describes a complete auto-generated HTML document based on the schema that is available for download from the Developer Services website at the following URL:  
<http://www.cisco.com/go/developersupport> (follow the **Supported Product** and **Getting Started** links).

**Note**

AXLAPI.wsdl file is built using AXL schema files- axl.xsd, axlsoap.xsd, AXLEnums.xsd and axlmessage.xsd. Various SOAP compilers, like .NET and Axis, have varying capabilities and compatibilities with WSDL; you may need to modify the AXLAPI.wsdl file to work with your specific compiler. Cisco does not test WSDL consumption with any specific compiler or version.

# Example XML Structure



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 means that all children of this element must appear in the XML in the order in which 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 HTTP Basic Authentication scheme; therefore, you must include the Authorization header in the HTTP 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==" represents the Base 64 encoding of "larry:curly and moe."



### Note

Cisco CallManager Database Layer Monitor service parameter 'EnableAXLAccessLevels' determines if AXL SOAP API will authorize or authenticate a CCMUser trying to use AXL API. If this parameter is set to True and MLA is enabled in Cisco CallManager Administration, MLA ReadOnly and SuperUserGroup access levels are enforced for users attempting to access the AXL API. For example, if a user is configured in the MLA ReadOnly user group, that user would not have write access in the AXL API, but could read the information; or if a user is configured in the MLA SuperUserGroup, that user would have read/write access in the AXL API. If the parameter 'EnableAXLAccessLevels' is set to False, all CCMUser accounts have full access to AXL SOAP.

## Data Encryption

If the user agent wants to encrypt the AXL SOAP message, the user agent must use HTTP SSL.

SSL does not function on the web server by default. Users can enable SSL through IIS configuration for CCMAPI virtual directory with this procedure: in IIS configuration, go to CCMAPI properties -> Directory Security tab -> Secure Communications section -> Edit -> and check the "Require secure channel (SSL)" check box. AXL requests can then get made by using the "https" protocol instead of "http".



### Note

Cisco does not recommend using this method to enable SSL as Cisco has not tested these settings.

## Integration Considerations and Inter-Operability

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

# 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/techsupport>

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](http://www.cisco.com/public/countries_languages.shtml)

## Product Documentation DVD

The Product Documentation DVD is a comprehensive library of technical product documentation on a portable medium. The DVD enables you to access multiple versions of installation, configuration, and command guides for Cisco hardware and software products. With the DVD, you have access to the same HTML documentation that is found on the Cisco website without being connected to the Internet. Certain products also have .PDF versions of the documentation available.

The Product Documentation DVD is available as a single unit or as a subscription. Registered Cisco.com users (Cisco direct customers) can order a Product Documentation DVD (product number DOC-DOCDVD= or DOC-DOCDVD=SUB) from Cisco Marketplace at this URL:

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

## Ordering Documentation

Registered Cisco.com users may order Cisco documentation at the Product Documentation Store in the Cisco Marketplace at this URL:

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

Nonregistered Cisco.com users can order technical documentation from 8:00 a.m. to 5:00 p.m. (0800 to 1700) PDT by calling 1 866 463-3487 in the United States and Canada, or elsewhere by calling 011 408 519-5055. You can also order documentation by e-mail at [tech-doc-store-mkpl@external.cisco.com](mailto:tech-doc-store-mkpl@external.cisco.com) or by fax at 1 408 519-5001 in the United States and Canada, or elsewhere at 011 408 519-5001.

## Documentation Feedback

You can rate and provide feedback about Cisco technical documents by completing the online feedback form that appears with the technical documents on Cisco.com.

You can submit comments about Cisco documentation 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](http://www.cisco.com/en/US/products/products_security_vulnerability_policy.html)

From this site, you will find information about how to:

- 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, security notices, and security responses for Cisco products is available at this URL:

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

To see security advisories, security notices, and security responses as they are updated in real time, you can subscribe to the Product Security Incident Response Team Really Simple Syndication (PSIRT RSS) feed. Information about how to subscribe to the PSIRT RSS feed is found at this URL:

[http://www.cisco.com/en/US/products/products\\_psirt\\_rss\\_feed.html](http://www.cisco.com/en/US/products/products_psirt_rss_feed.html)

## 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 have identified a vulnerability in a Cisco product, contact PSIRT:

- For Emergencies only—[security-alert@cisco.com](mailto:security-alert@cisco.com)

An emergency is either a condition in which a system is under active attack or a condition for which a severe and urgent security vulnerability should be reported. All other conditions are considered nonemergencies.

- For Nonemergencies—[psirt@cisco.com](mailto:psirt@cisco.com)

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

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

**Tip**

---

We encourage you to use Pretty Good Privacy (PGP) or a compatible product (for example, GnuPG) to encrypt any sensitive information that you send to Cisco. PSIRT can work with information that has been encrypted with PGP versions 2.x through 9.x.

Never use a revoked or an expired encryption key. The correct public key to use in your correspondence with PSIRT is the one linked in the Contact Summary section of the Security Vulnerability Policy page at this URL:

[http://www.cisco.com/en/US/products/products\\_security\\_vulnerability\\_policy.html](http://www.cisco.com/en/US/products/products_security_vulnerability_policy.html)

The link on this page has the current PGP key ID in use.

If you do not have or use PGP, contact PSIRT at the aforementioned e-mail addresses or phone numbers before sending any sensitive material to find other means of encrypting the data.

---

## Obtaining Technical Assistance

Cisco Technical Support provides 24-hour-a-day award-winning technical assistance. The Cisco Technical Support & Documentation website on Cisco.com features extensive online support resources. In addition, if you have a valid Cisco service contract, Cisco Technical Assistance Center (TAC) engineers provide telephone support. If you do not have a valid Cisco service contract, contact your reseller.

## Cisco Technical Support & Documentation Website

The Cisco Technical Support & Documentation 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, at this URL:

<http://www.cisco.com/techsupport>

Access to all tools on the Cisco Technical Support & Documentation 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>

**Note**

---

Use the Cisco Product Identification (CPI) tool to locate your product serial number before submitting a web or phone request for service. You can access the CPI tool from the Cisco Technical Support & Documentation website by clicking the **Tools & Resources** link under Documentation & Tools. Choose **Cisco Product Identification Tool** from the Alphabetical Index drop-down list, or click the **Cisco Product Identification Tool** link under Alerts & RMAs. The CPI tool offers three search options: by product ID or model name; by tree view; or for certain products, by copying and pasting **show** command output. Search results show an illustration of your product with the serial number label location highlighted. Locate the serial number label on your product and record the information before placing a service call.

---

## 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 provides recommended solutions. If your issue is not resolved using the recommended resources, your service request is assigned to a Cisco 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 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)—An existing 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 operations 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 the network is impaired, while 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.

- The *Cisco Product Quick Reference Guide* is a handy, compact reference tool that includes brief product overviews, key features, sample part numbers, and abbreviated technical specifications for many Cisco products that are sold through channel partners. It is updated twice a year and includes the latest Cisco offerings. To order and find out more about the Cisco Product Quick Reference Guide, go to this URL:

<http://www.cisco.com/go/guide>

- Cisco Marketplace provides a variety of Cisco books, reference guides, documentation, and logo merchandise. Visit Cisco Marketplace, the company store, at this URL:  
<http://www.cisco.com/go/marketplace/>
- *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>  
or view the digital edition at this URL:  
<http://cisoiq.texterity.com/cisoiq/sample/>
- *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>
- Networking products offered by Cisco Systems, as well as customer support services, can be obtained at this URL:  
<http://www.cisco.com/en/US/products/index.html>
- Networking Professionals Connection is an interactive website for networking professionals to share questions, suggestions, and information about networking products and technologies with Cisco experts and other networking professionals. Join a discussion at this URL:  
<http://www.cisco.com/discuss/networking>
- 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>

---

CCSP, CCVP, the Cisco Square Bridge logo, Follow Me Browsing, and StackWise are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn, and iQuick Study are service marks of Cisco Systems, Inc.; and Access Registrar, Aironet, BPX, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, FormShare, GigaDrive, GigaStack, HomeLink, Internet Quotient, IOS, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, LightStream, Linksys, MeetingPlace, MGX, the Networkers logo, Networking Academy, Network Registrar, *Packet*, PIX, Post-Routing, Pre-Routing, ProConnect, RateMUX, ScriptShare, SlideCast, SMARTnet, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0601R)

*AXL Programming Guide for Cisco Unified CallManager 4.2(3)*  
Copyright © 2006. Cisco Systems, Inc. All rights reserved.

