



Cisco CallManager 4.1(2) AXL Programming Guide

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

http://www.cisco.com/warp/public/570/avvid/voice_ip/axl_soap/axl_api_down.html

This manual contains the following sections:

- [Introduction, page 2](#)
- [Target Audience for this Guide, page 2](#)
- [New and Changed Information, page 2](#)
- [AXL API, page 4](#)
- [Examples of AXL Requests, page 7](#)
- [Throttling of Requests, page 12](#)
- [The AXL Schema Documentation, page 12](#)
- [Example XML Structure, page 13](#)
- [Authentication, page 13](#)
- [Data Encryption, page 14](#)
- [Integration considerations and Inter-operability, page 14](#)
- [Obtaining Documentation, page 14](#)
- [Documentation Feedback, page 15](#)
- [Obtaining Technical Assistance, page 15](#)
- [Obtaining Additional Publications and Information, page 16](#)

Introduction

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

Target Audience for this Guide

This programming guide is designed for fairly experienced developers who would like access to one or more of the following items:

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

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

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

In addition, the 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, please refer to <http://www.w3.org/TR/xmlschema-0/>.

**Caution**

The AXL API gives enormous power to developers to modify the Cisco CallManager system database. The developer must take caution when using AXL, since each API call impacts the system. Abuse 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.

New and Changed Information

This section describes the new or changed API calls for Cisco CallManager Release 4.1(2).

Added API Calls

- addTimePeriod
- updateTimePeriod

- removeTimePeriod
- getTimePeriod
- addTimeSchedule
- updateTimeSchedule
- removeTimeSchedule
- getTimeSchedule
- addCMCInfo
- updateCMCInfo
- removeCMCInfo
- getCMCInfo
- addFACInfo
- updateFACInfo
- removeFACInfo
- getFACInfo

Changed API Calls

These changes may require changes to existing user code that makes use of the following:

- addPhone
- updatePhone
- getPhone
- addLine
- updateLine
- addUser
- removeUser
- updateUser
- getUser
- addDeviceProfile
- updateDeviceProfile
- getDeviceProfile
- addRoutePattern
- updateRoutePattern
- getRoutePattern
- addRouteList
- updateRouteList
- getRouteList
- addGatewayEndpoint
- updateGatewayEndpoint

- getGatewayEndpoint
- addH323Trunk
- updateH323Trunk
- removeH323Trunk
- getH323Trunk
- addHuntPilot
- updateHuntPilot
- removeHuntPilot
- getHuntPilot
- addProcessNode
- updateProcessNode
- removeProcessNode
- getProcessNode
- listAllProcessNodes
- listProcessNodesByService
- addRoutePartition
- updateRoutePartition
- removeRoutePartition
- getRoutePartition
- addH323Gateway
- updateH323Gateway
- removeH323Gateway
- getH323Gateway
- addH323Phone
- updateH323Phone
- removeH323Phone
- getH323Phone
- addHuntList
- updateHuntList
- removeHuntList
- getHuntList

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" has been appended.

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

If an error occurs, then an XML error structure is returned wrapped inside of a SOAP Fault structure (see [“AXL Error Codes” section on page 5](#)).

AXL Methods

The Server Impact column is a sample measuring of the time (in milliseconds) it takes for the AXL server to handle the request. The tests were performed on a 7835-1000 MCS server. These figures should only be used as a guideline, and can drastically change due to cluster configuration, hardware configuration, and the size of the database.

The database used for these tests had the following entries:

- 1 Conference Bridge
- 2 CTI Route Points
- 2 Media Termination Points
- 1 Music On Hold
- 61 Phones
- 1 Route List
- 2 Call Pick Up Groups
- 2 Call Parks
- 1 Conference
- 74 Directory Numbers
- 1 Route Pattern
- All Cisco services running

AXL Error Codes

If an exception occurs on the server, or if any other error occurs during the processing of an AXL request, then 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/" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Client</faultcode>
      <faultstring>Device not found with name SEP003094C39708.</faultstring>
      <detail xmlns:axl="http://www.cisco.com/AXL/1.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.cisco.com/AXL/1.0
          http://myhost/CCMApi/AXL/V1/axlsoap.xsd">
        <axl:error sequence="1234">
          <code>0</code>
          <message>
            <![CDATA[
              Device not found with name SEP003094C39708.
            ]]>
          </message>
          <request>doDeviceLogin</request>
        </axl:error>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Error codes are included in the <detail> element of a SOAP Fault. The errors are represented by the axl:Error element. If a response to a request contains an <error> element, the user agent can determine the cause of the error by looking at the sub-elements of the <error> tag.

The following list describes the <error> element.

code

The <code> element is a numerical value that is used by the user agent to find out what type of error has occurred. The error codes are as follows:

Error Code	Description
Less than 5000	These are errors that directly correspond to DBL Exception error codes. Please 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 be 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.

Error Code	Description
5004	AXL Unavailable Exception—This error occurs if the AXL service is too busy to handle the request at that time. The request should be sent 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>, then this error is returned. These errors are always caused by malformed requests that do not adhere to the latest AXL Schema.

message

The <message> element is provided so that the user agent gets a detailed error message explaining the error.

request

The <request> element is provided so that the user agent can determine what type of request generated this error. This element is optional; therefore it may not always appear.

Examples of AXL Requests

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

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

The first header identifies 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 being sent to AXL is XML. If this header is not found then an HTTP 415 error is returned to the client.

- Authorization: Basic <some Base 64 encoded string>

The third header is the Base64 encoding of the user name and password for the administrator of the AXL Server. If authentication of the user fails, then an HTTP 401 Access Denied error is returned to the client.

- content-length: <a positive integer>

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

**Note**

Currently, the content length cannot exceed 40 kilobytes. If a request of more than 40 kilobytes is received, then an HTTP 413 error message is returned.

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 will be used in the code examples displayed in the following sections. This is an example of 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">
      <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 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 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://gkar.cisco.com/schema/axlsoap.xsd\\" sequence=\\"1234\\""> \
<phoneName>SEP2222222222245</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 HTTP header
strHTTPHeader = "POST /CCMApi/AXL/V1/soapisapi.dll\r\n \
Host: localhost:80\r\n \
Authorization: Basic bGFYcnk6Y3VybHkgYW5kIGlvZQ==\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;
        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 running on the local system (localhost). It then reads the response back, outputting 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 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://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 CallManager database can adversely affect system performance; therefore, the system administrator is capable of controlling how many AXL requests are allowed to update the database per minute. This value is controlled via 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 HTTP 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 are considered "Reads" and do not count against the "MaxAXLWritesPerMinute" service parameter. All other AXL requests not in the following list count against the service parameter:

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

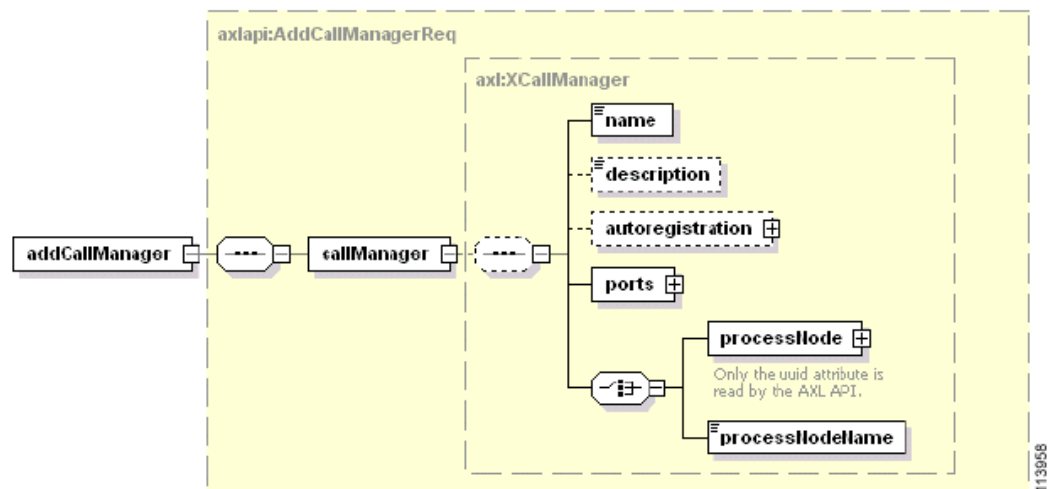
The AXL Schema Documentation

The complete AXL schema (including details of all requests, responses, XML objects, data types, etc.) is encapsulated in the following four files found on the Cisco CallManager server: axlsoap.xsd, axl.xsd, axlmessage.xsd, and AXLEnums.xsd. The default path is: **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 describes a complete auto-generated HTML document based on the schema that is available for download from the Developer Services web site at the following URL:
<http://www.cisco.com/go/developersupport> (follow the **Supported Product** link).

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 is required to 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 is required to appear in an XML instance document.
	Optional Element	An optional element is not required 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 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

Anonymous access to the AXL SOAP service should be deactivated 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 wishes 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 Base 64 encoding of "larry:curly and moe".

Data Encryption

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

SSL is not functional on the web server by default. The customer must request and install a SSL Certificate from a certified authority such as VeriSign. Once the SSL Certificate has been installed on the web server, AXL requests can be made using the "https" protocol in lieu of "http".

Integration considerations and Inter-operability

The AXL API gives much power to developers to modify the Cisco CallManager system database. The developer must take caution when using AXL, since each API call impacts the system. Abuse 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 of the 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 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/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

Ordering 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.

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>

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>

CCSP, the Cisco Square Bridge logo, Cisco Unity, Follow Me Browsing, FormShare, 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 Aironet, ASIST, 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, Empowering the Internet Generation, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, 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, Registrar, ScriptShare, SlideCast, SMARTnet, StrataView Plus, SwitchProbe, TeleRouter, The Fastest Way to Increase Your Internet Quotient, TransPath, and VCO 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. (0406R)

Copyright © 2004 Cisco Systems, Inc. All rights reserved.

 Printed in the USA on recycled paper containing 10% postconsumer waste.