Cisco CallManager API Troubleshooting Guide

Last release date:
02/9/2004

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

CONTENTS

Preface

Audience

This document is a reference guide to assist end-users, customers, developers, and support staff in understandig and troubleshooting various Cisco CallManager-related Application Programming Interfaces (APIs) and interfaces.

Structure of This Guide

This guide contains the following sections:

- General Troubleshooting Procedures, page 1—Provides suggestions for addressing common problems, help on opening a developer support case, and a discussion of coordinated support.
- TAPI Troubleshooting, page 9—Provides an overview of the Telephony Applications Programming Interface (TAPI), a discussion of application architecture, a post-installation checklist, related troubleshooting tools, and tips on obtaining error reports.
- JTAPI Troubleshooting, page 21—Provides an overview of the Java Telephony Applications Programming Interface (JTAPI), a discussion of application architecture, a post-installation checklist, related troubleshooting tools, and tips on obtaining error reports.
- XML Services Troubleshooting, page 27—Provides an overview of Extensible Markup Language (XML) services, a discussion of application architecture, a discussion of available troubleshooting tools, and tips on obtaining error reports.
- AXL Troubleshooting, page 35—Provides an overview of the AVVID XML Layer (AXL), a discussion of application architecture, a post-installation checklist, related troubleshooting tools, and tips on obtaining error reports.
- SCCP (Skinny) Phone/Endpoint Troubleshooting, page 45—Provides an overview of Skinny, a brief example application, a discussion of available troubleshooting tools, and tips on obtaining error reports.
- Cisco CallManager Express and Survivable Remote Site Telephony Troubleshooting, page 53—Provides an overview of application architecture, a post-installation checklist, and sample debug traces.
- Glossary—Presents an alphabetical listing of common terms used throughout this document.

Related Documents

Document	Title or URL
Microsoft Windows Telephony Features with TAPI 2.1	http://www.microsoft.com/ntserver/techresources/commnet/tele/tapi21.asp
Java Telephony API	http://java.sun.com/products/jtapi/
Cisco IP Phone Services	http://www.cisco.com/univercd/cc/td/doc/product/voice/c_callmg/3_3/sys_ad/3_3_3/ccmsys/a07phsvc.htm
Cisco IP Phone Services Configuration	http://www.cisco.com/univercd/cc/td/doc/product/voice/c_callmg/3_3/sys_ad/3_3_3/ccmcfg/b05phsrv.htm
Cisco CallManager Troubleshooting Guides	http://www.cisco.com/univercd/cc/td/doc/product/voice/c_callmg
Developing Cisco IP Phone Services	Publication by Darrick Deel, Mark Nelson, and Anne Smith. Published by Cisco Press, ISBN 1=58705-060-9
Cisco IP Phone Services Application Development Notes	http://www.cisco.com/univercd/cc/td/doc/product/voice/vpdd/cdd/3_3/3_3_4/ipphs/index.htm
Cisco IP Phone Services XML Schema	http://www.cisco.com/univercd/cc/td/doc/product/voice/vpdd/cdd/3_3/3_3_4/ipphs/ip334apb.htm

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 on the World Wide Web at this URL:

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

You can access the Cisco website at this URL:

http://www.cisco.com

International Cisco websites can be accessed from 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 submit e-mail 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, the Cisco Technical Assistance Center (TAC) provides 24-hour-a-day, award-winning technical support services, online and over the phone. Cisco.com features the Cisco TAC website as an online starting point for technical assistance. If you do not hold a valid Cisco service contract, please contact your reseller.

Coordinated Support

The following is a list of questions and related answers associated with Coordinated Support:

Q. What is Coordinated Support?

A. Coordinated Support is a comprehensive service offered to Cisco AVVID Partner Program participants, designed to provide seamless technical support to joint customers with valid maintenance agreements for their Cisco AVVID solutions.

Q. How is Coordinated Support organized?

A. There are two levels of Coordinated Support: Type I and Type II.

Type I is provided for verified interoperable products in solution categories whose failure impact on network operations is minimal. These are the products whose failure might result in a Priority 3 or a Priority 4 technical support call. Partners whose products fall into this category are still expected to be able to provide Cisco with a high level of support in the event that Cisco needs assistance on a Priority 1 or Priority 2 or a Cisco Critical Accounts Program (CAP) case.

Type II is provided for verified interoperable products in solution categories that potentially have a critical impact on network operations if a problem arises. Partners will be required to adhere to a higher support standard in order to ensure that both Cisco and the partner can adequately respond and meet the expectations of our mutual customers.

Q. Why are there two levels of Coordinated Support?

A. Type I support has been created for those partners who do not rely heavily on the Cisco development application protocol interfaces (APIs) and protocols and whose product failure will have minimal impact to network operations. Type II support was created for partners whose products have critical impact on the network, should these products malfunction.

Q. Does Coordinated Support require the hot or warm hand-off of a customer call?

A. Hot or warm hand-off of customer support calls is not required under Coordinated Support; however, for critical issues or in cases where the location of the problem remains unclear after troubleshooting, the Cisco support engineer or the partner's support engineer may choose to work together to resolve the support issue.

Q. Where can I find more information about the Cisco AVVID Coordinated Support processes and procedures?

A. The Cisco AVVID Partner Program Coordinated Support Manual contains detailed processes and procedures and is located at
http://www.cisco.com/warp/public/cc/so/cuso/epso/avpnpg/appcs_ds.htm

Cisco TAC Website

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

http://www.cisco.com/tac

Accessing all the tools on the Cisco TAC website requires a Cisco.com user ID and password. If you have a valid service contract but do not have a login ID or password, register at this URL:

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

Opening a TAC Case

Using the online TAC Case Open Tool is the fastest way to open P3 and P4 cases. (P3 and P4 cases are those in which your network is minimally impaired or for which you require product information.) After you describe your situation, the TAC Case Open Tool automatically recommends resources for an immediate solution. If your issue is not resolved using the recommended resources, your case will be assigned to a Cisco TAC engineer. The online TAC Case Open Tool is located at this URL:

http://www.cisco.com/tac/caseopen

For P1 or P2 cases (P1 and P2 cases are those in which your production network is down or severely degraded) or if you do not have Internet access, contact Cisco TAC by telephone. Cisco TAC engineers are assigned immediately to P1 and P2 cases to help keep your business operations running smoothly.

To open a case 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 listing of Cisco TAC contacts, go to this URL:

http://www.cisco.com/warp/public/687/Directory/DirTAC.shtml

TAC Case Priority Definitions

To ensure that all cases are reported in a standard format, Cisco has established case priority definitions.

Priority 1 (P1)—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.

Priority 2 (P2)—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.

Priority 3 (P3)—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.

Priority 4 (P4)—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. Go to this URL to visit the company store:

  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 online at this URL:

  http://www.ciscopress.com

- *Packet* magazine is the Cisco quarterly publication that provides the latest networking trends, technology breakthroughs, and Cisco products and solutions to help industry professionals get the most from their networking investment. Included are networking deployment and troubleshooting tips, configuration examples, customer case studies, tutorials and training, certification information, and links to numerous in-depth online resources. You can access Packet magazine at this URL:

  http://www.cisco.com/packet

- *iQ Magazine* is the Cisco bimonthly publication that delivers the latest information about Internet business strategies for executives. 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

- Training—Cisco offers world-class networking training. Current offerings in network training are listed at this URL:

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

General Troubleshooting Procedures

This chapter contains the following information:

- Common Problems, page 1
- Reporting Problems, page 2
- Opening a Developer Support Case, page 6

Common Problems

This section provides information on resolving common problems seen by both end customers and developers.

Customers

Verify the following initial-setup information for your application programming interface (API):

- TAPI—Postinstallation Checklist, page 12.
- JTAPI—Postinstallation Checklist, page 22.
- XML—Troubleshooting Checklist, page 31
- AXL—Postinstallation Checklist, page 36
- Skinny—Troubleshooting Tools, page 46
- CCME/SRST—PostInstallation Checklist for TAPILite, page 54

If the problem persists, open a TAC case (see “Opening a Developer Support Case” on page 6).

Developers

See the complete listing of FAQs for your API. Follow the "Product FAQs" link. (Go to Supported Products and select the interface from the dropdown menus). Also, see the Developer Services FAQ Central page at http://www.cisco.com/cgi-bin/front.x/cse/DSFC/DSFCHome.pl. If your problem is still not fixed, open a Developer Support case from the link at http://www.cisco.com/cgi-bin/front.x/case_tools/caseOpen.pl. (See “Opening a Developer Support Case” on page 6.)

Reporting Problems

If you have simple questions that do not require more than 15 minutes to resolve, send those to the appropriate Developer Support alias:

- TAPI and JTAPI—tapi_jtapi-sdp@cisco.com
- AXL—axl-api-sdp@cisco.com
- Skinny—endpoints-sdp@cisco.com
- Cisco CallManager Express/SRST—ccme-srst-api-sdp@cisco.com

For issues involving the integration or deployment of your application with Cisco CallManager, collect all the recommended traces (see the “Error Reporting” section in the appropriate Troubleshooting chapter) in addition to the CTI, SDL, and CCM traces before opening a Developer Support case. For instructions on opening a Developer Support case, see “Opening a Developer Support Case” on page 6.

Use the Trace Collection Tool, a client-side plugin, to collect and zip various Cisco CallManager service traces and/or other Cisco CallManager log files in the form of a single/multiple zip file(s). For more information on using this tool, see the “Trace Configuration Tool” section in the *Cisco CallManager Serviceability System Guide*.

To determine the appropriate reporting steps, check the following sections:

- Reporting Problems for Cisco CallManager Release 3.3 or Older, page 2
- Reporting Problems for Cisco CallManager Release 4.0, page 4

Reporting Problems for Cisco CallManager Release 3.3 or Older

The steps to collect these traces are as follows:

Step 1 Make sure the SDL (SDLxxx_100_xxxxxx files) trace settings are set appropriately. Under Service->Service Parameters, select the IP address of the server. Then select the Cisco CallManager (CM) service from the list of services. Check or set the following CM service parameters from the CM administration:

Note When changing parameters, do not forget to click the **update** button to save any changes.

- **SdlTraceDataFlags**—Set to 0x00000110.
- **SdlTraceFlag**—Set to T.
- **SdlTraceMaxLines**—Set to between 10000 and 20000.
- **SdlTraceTotalNumFiles**—Set to at least 100. If you are running high traffic, this number should be increased significantly. The best thing to do is after running for a while, see how much time is being captured by looking at the date/time of the SDL files. If the files are being overwritten before you would be able to save it off, increase the SdlTraceTotalNumFiles appropriately.
- **SdlTraceTypeFlags**—Set to 0x8000EB15.

Step 2 Make sure the SDL (SDLxxx_200_xxxxxx files) trace settings are set appropriately. Under *Service->Service Parameters*, select the IP address of the server. Then select the Cisco CTIManager service from the list of services. Check or set the following Cisco CTIManager service parameters from the CM administration:

Note When changing parameters, do not forget to click the **update** button to save any changes.

- **SdlTraceDataFlags**—Set to 0x00000110.
- **SdlTraceFlag**—Set to T.
- **SdlTraceMaxLines**—Set to between 10000 and 20000.
- **SdlTraceTotalNumFiles**—Set to at least 100. If you are running high traffic, this number should be increased significantly. The best thing to do is, after running for awhile, see how much time is being captured by looking at the date/time of the SDL files. If the files are being overwritten before you are able to save them off, increase the SdlTraceTotalNumFiles appropriately.
- **SdlTraceTypeFlags**—Set to 0x0000CB15.

Step 3 Make sure the Cisco CallManager serviceability (ccmxxxxxxxx files) trace settings are set up for the Cisco CallManager service. In the CM administration under Application->Cisco CallManager Serviceability, select **Trace->Configuration**. Select the IP address of the server from the list of servers. Select **Cisco CallManager** from the list of Configured Services. The **Trace On** check box should be checked.

- Under Trace Filter Settings:
  - The Debug Trace Level should be set to **Detailed**.
  - Under Cisco CallManager Trace Fields, make sure the check boxes for the trace related to the problem are checked. For example, if the problem has to do with a call made through a DT-24+ gateway, check the check box labeled **Enable DT-24+/DE-30+ Trace**.
- Under Trace Output Settings:
  - The **Enable File Trace Log** check box should be checked.
  - The Filename edit box should contain a valid path and filename.
  - The default for Maximum No. of Files is 250.
  - The default for Maximum No. of Lines per File is 10000.
  - The default for Maximum No. of Minutes per File is 1440.
- As with the SDL traces, after the test has been running a while, check to see if the Cisco CallManager trace files are being overwritten too soon (before they can be copied). If they are, increase the Maximum No. of Files parameter appropriately.

Step 4 Make sure the Cisco CTIManager serviceability (ccmxxxxxxxx files) trace settings are setup for the Cisco CTIManager service. In the CM administration under Application->Cisco CallManager Serviceability, select **Trace->Configuration**. Select the IP address of the server from the list of servers. Select **Cisco CTIManager** from the list of Configured Services. The **Trace On** check box should be checked.

- Under Trace Filter Settings:
  - The Debug Trace Level should be set to **Detailed**.
  - The **Cisco CTIManager Trace Fields** check box should be checked.
  - The **Enable All Trace** check box should be checked.
- Under Trace Output Settings:
  - The **Enable File Trace Log** check box should be checked.
  - The Filename edit box should contain a valid path and filename.

  - The default for Maximum No. of Files is 250.
  - The default for Maximum No. of Lines per File is 10000.
  - The default for Maximum No. of Minutes per File is 1440.
- As with the SDL traces, after the test has been running awhile, check to see if the Cisco CallManager trace files are being overwritten too soon (before they can be copied). If they are, increase the Maximum No. of Files parameter appropriately.

Step 5 If settings were changed for Cisco CTIManager, you must restart the service for the settings to take effect.

Step 6 When a Cisco CallManager problem happens, save off the SDL, CCM, and other related trace files (JTAPI, app traces, etc.).

Reporting Problems for Cisco CallManager Release 4.0

Use the following steps to trace the problem:

Make Sure the SDL (SDLxxx_100_xxxxxx files) Trace Settings are Set Appropriately

1. Under Service->Service Parameters, select the IP address of the server.
2. Select the **Cisco CallManager (CM) service** from the list of services.
3. Check or set the following CM service parameters from the SDL Trace section:

Note When changing parameters, do not forget to click the **update** button to save any changes.

   - **SdlTraceDataFlags**—Set to 0x00000110.
   - **SdlTraceFlag**—Set to True.
   - **SdlTraceTypeFlags**—Set to 0x8000EB15.
4. In the CM administration under Application->Cisco CallManager Serviceability, select Trace->Configuration.
5. Select the IP address of the server from the list of servers.
6. Select **Cisco CallManager** from the list of Configured Services.
7. Click the SDL Configuration link. Scroll down to the Trace Output Settings section. Update the following service parameters as given below:
   - **Trace Directory Path**—Set the path where you want the log files to be created.
   - **SdlTraceMaxLines**—Set to between 10000 and 20000.
   - **SdlTraceTotalNumFiles**—Set it to at least 100. If you are running high traffic, this number should be increased significantly. The best thing to do is after running for a while, see how much time is being captured by looking at the date/time of the SDL files. If the files are being overwritten before you would be able to save it off, increase the SdlTraceTotalNumFiles appropriately.

Make Sure the SDL (SDLxxx_200_xxxxxx files) Trace Settings are Set Appropriately.

1. Under *Service->Service Parameters*, select the IP address of the server.

2. Select the **Cisco CTIManager service** from the list of services.
3. Click on the **Advanced** button and scroll down to the SDL Trace Parameters group. Update the following values :

Note When changing parameters, don't forget to click the **update** button to save any changes.

- **SdlTraceDataFlags**—Set to 0x00000110.
- **SdlTraceFlag**—Set to True.
- **SdlTraceTypeFlags**—Change the default setting of 0x00D0CB15 to 0x00F0CB15.

4. In the CM administration under Application->Cisco CallManager Serviceability, select Trace->Configuration.
5. Select the IP address of the server from the list of servers.
6. Select **Cisco CTIManager** from the list of Configured Services.
7. Click **SDL Configuration** link. Scroll down to the Trace Output Settings section. Update the following service parameters as given below.

Note The following service parameters are read-only on the SDL Trace Parameters page. These values can be set on the Cisco CallManager Serviceability page.

- **Trace Directory Path**—Set the path where you want the log files to be created.
- **SdlTraceMaxLines**—Set to between 10000 and 20000.
- **SdlTraceTotalNumFiles**—Set to at least 100. If you are running high traffic, this number should be increased significantly. The best thing to do is, after running for awhile, see how much time is being captured by looking at the date/time of the SDL files. If the files are being overwritten before you are able to save them off, increase the SdlTraceTotalNumFiles appropriately.

Make Sure the Cisco CallManager Serviceability (ccmxxxxxxxx files) Trace Settings are Set Up for the Cisco CallManager Service.

1. In the CM administration under Application->Cisco CallManager Serviceability, select **Trace->Configuration**.
2. Select the IP address of the server from the list of servers.
3. Select **Cisco CallManager** from the list of Configured Services. The **Trace On** check box should be checked.
   - Under Trace Filter Settings:
     – The Debug Trace Level should be set to **Detailed**.
     – Under Cisco CallManager Trace Fields, make sure the check boxes for the trace related to the problem are checked. For example, if the problem has to do with a call made through a DT-24+ gateway, check the check box labeled **Enable DT-24+/DE-30+ Trace**.
   - Under Trace Output Settings:
     – The **Enable File Trace Log** check box should be checked.
     – The Filename edit box should contain a valid path and filename.
     – The default for Maximum No. of Files is 250.

    - The default for Maximum No. of Lines per File is 10000.
    - The default for Maximum No. of Minutes per File is 1440.
  - As with the SDL traces, after the test has been running a while, check to see if the Cisco CallManager trace files are being overwritten too soon (before they can be copied). If they are, increase the Maximum No. of Files parameter appropriately.

4. Make sure the Cisco CTIManager serviceability (ccmxxxxxxxxx files) trace settings are setup for the Cisco CTIManager service.
5. In the CM administration under *Application->Cisco CallManager Serviceability*, select **Trace->Configuration**.
6. Select the IP address of the server from the list of servers.
7. Select **Cisco CTIManager** from the list of Configured Services. The **Trace On** check box should be checked.
   - Under Trace Filter Settings:
     - The Debug Trace Level should be set to **Detailed**.
     - The **Cisco CTIManager Trace Fields** check box should be checked.
     - The **Enable All Trace** check box should be checked.
   - Under Trace Output Settings:
     - The **Enable File Trace Log** check box should be checked.
     - The Filename edit box should contain a valid path and filename.
     - The default for Maximum No. of Files is 250.
     - The default fo*r* Maximum No. of Lines per File is 10000.
     - The default for Maximum No. of Minutes per File is 1440.
   - As with the SDL traces, after the test has been running a while, check to see if the Cisco CallManager trace files are being overwritten too soon (before they can be copied). If they are, increase the Maximum No. of Files parameter appropriately.
8. If settings were changed for Cisco CTIManager, you must restart the service for the settings to take effect.

Note If a Cisco CallManager problem happens, save off the SDL, CCM, and other related trace files (JTAPI, app traces, etc.).

Opening a Developer Support Case

Follow these instructions to open a Cisco Developer Support Case:

Developer Support Web Page

1. Enter your Developer Support Contract Number from the cover e-mail: ____________

Note Each development team member must register on Cisco.com with this contract ID before they can open a support case.

If you already have a Cisco.com profile, send an e-mail to cco-team@cisco.com and request to have this contract number added to your user id profile. Allow 24 hours for the new information to be populated.

2. Register by using the following procedure:

   a. From www.cisco.com, click "Register" on the top menu bar.

   b. Select “1a - Cisco Service Contract Owner.” Do not select any additional privileges for this contract.

   c. Enter your contract number. No verification key is required—leave blank.

   d. Complete the registration form *with your company information* and create your userid and password associated with this contract number.

   Note You will receive a confirmation e-mail.

3. Open a Support Case

   a. Login to www.cisco.com with the user id and password you created.

Note If you cannot remember your userid and/or password, contact the Cisco Customer Response Center (see below).

   b. Under the **Service and Support** section, select **Technical Support Help—Cisco TAC**.

   c. Select **Contact TAC/Check Status**.

   d. Under *Manage a TAC Case*, select **Open a TAC Case**.

   e. Verify that the *Contract* field contains your Developer Support contract number.

   f. In *Technology* field 1, select **Other**.

   g. In *Technology* field 2, select **Developer Support Program**.

   h. Under *Problem Summary*, select the appropriate product.

   i. Enter a complete problem description and attach any documentation to support your case. No other fields need to be completed.

   j. Click **Submit**. Your case is automatically opened and an e-mail confirmation is sent to you.

Call Response Center

If you are having problems opening a Case on Cisco.com, call the Cisco Customer Response Center (CRC) at 1 877 841-3971 (408 525-4810) and provide your Contract Number. You will be asked for the required information to open a case, then the case will be directed to the proper support team.

Note The CRC staff are not technical support personnel so cannot address problems themselves. They will route your call based on the information you provide them.

Lost Contract Numbers: If you lose or misplace your unique identification number, contact the Cisco Customer Response Center and identify yourself as a member of the Cisco Developer Support Program.

Table 1 provides a general guideline for case response times. Case response times are based on standard business hours where the support team is located.

Table 1 Case Response Guideline

	Priority 1	Priority 2	Priority 3	Priority 4
Initial Response	4 Hours	1 Day	1.5 Days	2 days

TAPI Troubleshooting

This chapter contains the following topics:

Overview

TAPI

Telephony Applications Programming Interface (TAPI) is part of the Microsoft Windows Open System Architecture and provides developers with the ability to create telephony applications.

TAPI is:

- An open industry standard
- Defined with considerable and ongoing input from the worldwide telephony and computing community
- Switch-fabric independent

TAPI-compatible applications can run on a wide variety of PC and telephony hardware and can support a variety of network services.

TAPI supports many telephony features, allowing developers to telephone-enable virtually any general-purpose application. TAPI also supports Unicode, making it easier to effect global applications. With Active Controls provided by a variety of vendors, corporate developers can put together telephony applications using tools they are already familiar with.

For more information about TAPI refer to http://www.microsoft.com/ntserver/techresources/commnet/tele/tapi21.asp.

TSP

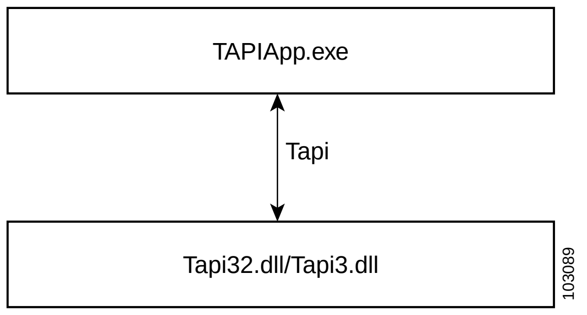
Telephony Service Provider (TSP) is a dynamic-link library (DLL) that provides device-specific controls for communications processing. TSP provides an abstraction layer between TAPI applications and the underlying hardware and transport protocols.

Architecture

TAPI applications reside in their own process space and communicate with Tapi32.dll or Tapi3.dll using TAPI. See Figure 1.

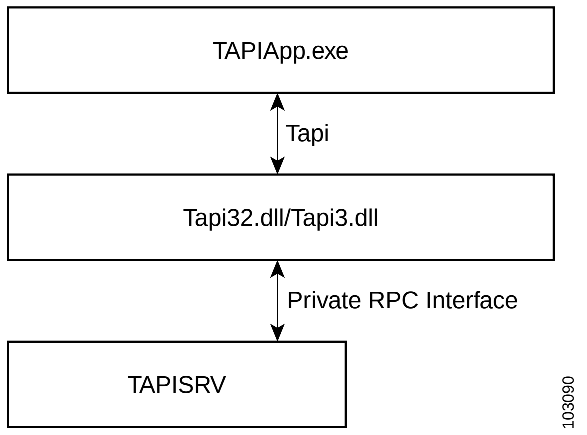
Note Tapi32.dll and Tapi3.dll are DLLs provided by Microsoft.

Figure 1 Telephony Application PC



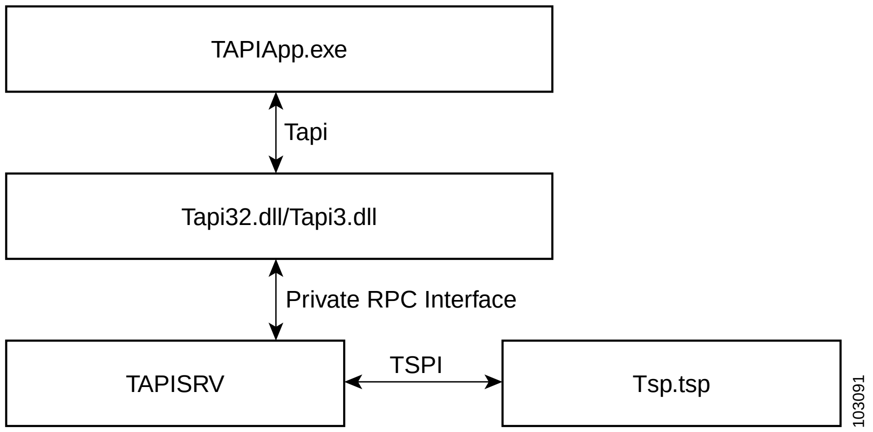
Tapi32.dll and Tapi3.dll communicate with the TAPISRV process using a private RPC interface. TAPISRV is a Microsoft process. In Windows 95, 98, and NT, this process is names TAPISRV.EXE. In Windows 2000, this process runs under the SVCHOST.EXE process. See Figure 2.

Figure 2 Telephony Application PC



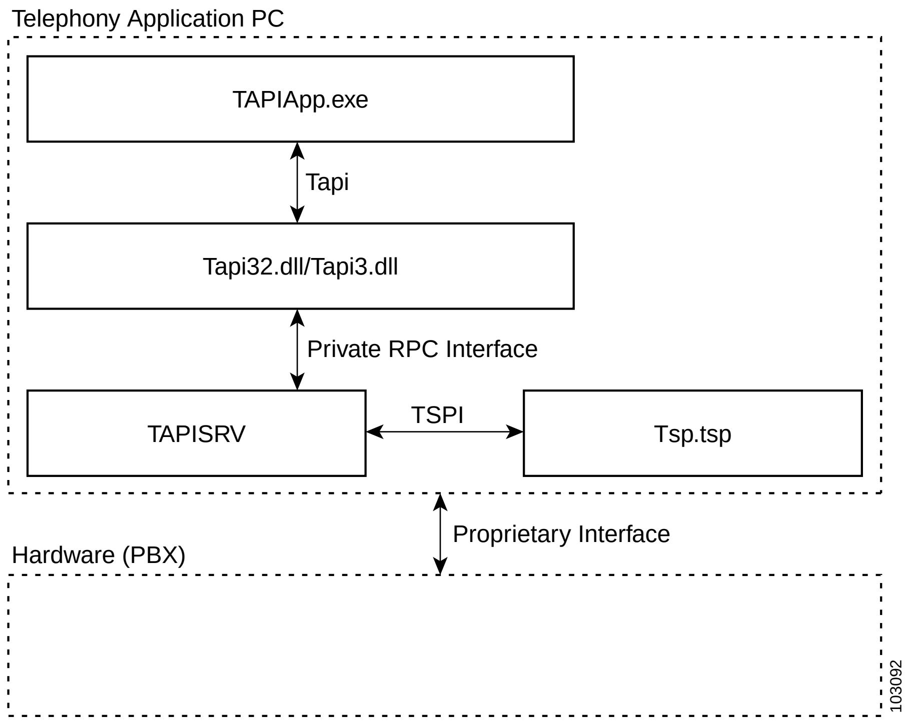
TAPISRV communicates with the TSP using a Microsoft-defined interface known as TSPI. The TSP is a DLL that is dynamically linked to the TAPISRV process. Each PBX vendor provides its own TSP. See Figure 3.

Figure 3 Telephony Application PC



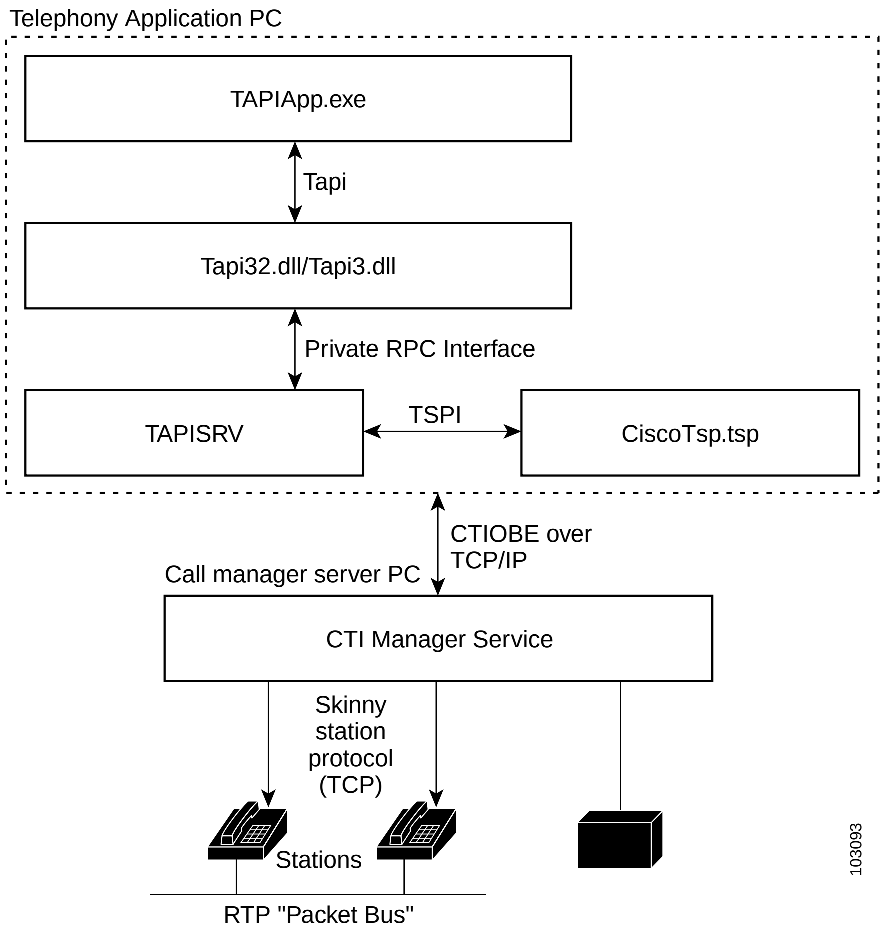
The TSP communicates with the underlying telephony hardware, such as a PBX, using a proprietary interface. The telephony hardware may be on the same PC or on a different PC. See Figure 4.

Figure 4 Telephony Application PC



Cisco provides a TSP called CiscoTSP. CiscoTSP communicates with the Cisco CallManager using a proprietary interface known as CTIQBE. CiscoTSP allows customers to create customized IP telephony applications for the Cisco CallManager. See Figure 5.

Figure 5 Telephony Application PC



Postinstallation Checklist

Some problems may be the result of improper installation. Use the following checklist to verify proper installation before proceeding with the troubleshooting process:

____ Check that the CTI Manager Location is configured properly.

____ Check to see if the CTI Manager is up and running.

____ Check to see that the network connection is up between the TSP and CTI Manager machines.

____ Check to see if the Cisco CallManager DC Directory Service is up and running.

____ Check that the Username and Password that is configured in the TSP matches the Username and Password in the Cisco CallManager Directory.

____ Make sure that the Enable CTI Application Use flag has been checked in the Directory Administration on the Cisco CallManager for the TSP user.

____ Check that devices have been properly associated with the user in the Cisco CallManager Directory.

____ Check to see if the CTI Connection Limit has been reached.

____ For non-SoftPhone machines, ensure the Cisco Wave Drivers have been installed.

____ Use the tools in "Troubleshooting Tools" on page 13 to verify the TSP is working properly.

____ Ensure the CiscoTSP version is compatible with the version of Cisco CallManager. This can be verified by checking the Cisco CallManager Compatibility Matrix. Refer to the *Defaul Loads* section on the Cisco Connection Online website at http://www.cisco.com/univercd/cc/td/doc/product/voice/c_callmg/ccmcomp.htm#MinimumVersions

Troubleshooting Tools

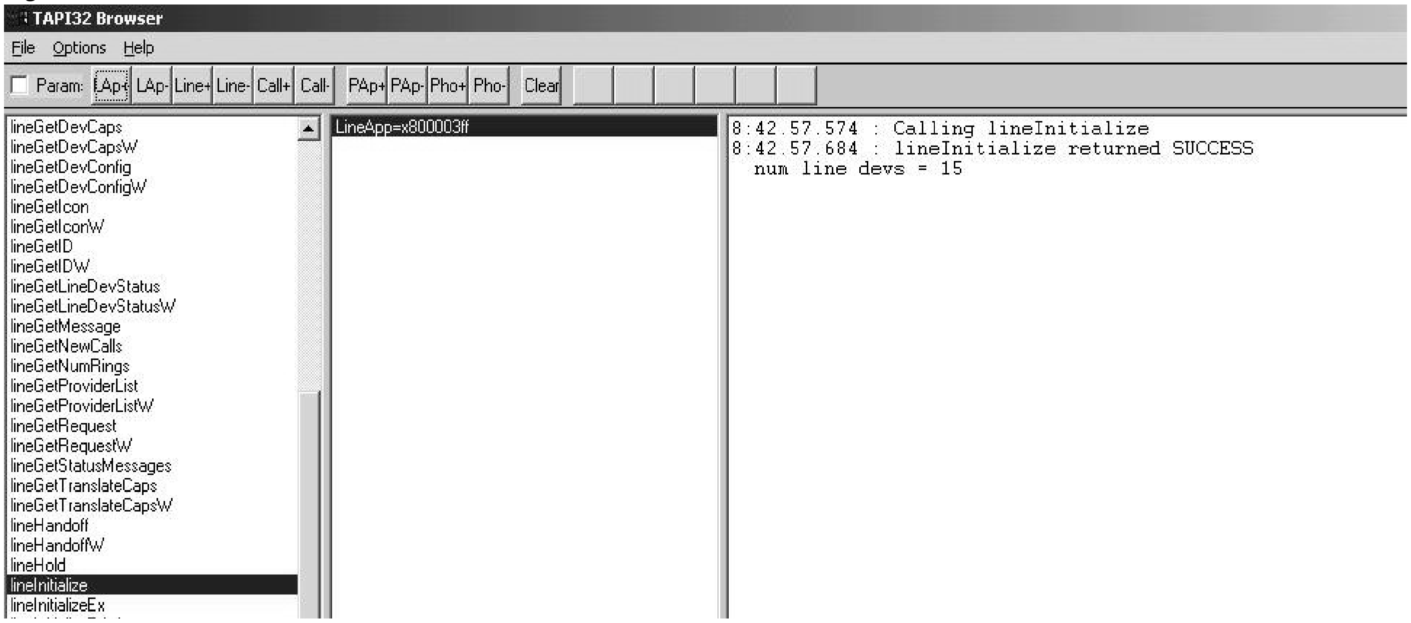
This section describes available troubleshooting tools.

TAPI Browser

The TAPI Browser does not come standard with the Window operating system. It can be obtained from the Microsoft MSDN website. The TAPI Browser can be used to initialize TAPI. Its main use is by TAPI developers to test their TAPI implementation and to verify that the TSP is operational. See Figure 6.

To do this, bring up the TAPI Browser (TB20.exe), then double-click **lineInitializeEx** in the leftmost window. If *num line devs* returns the expected number of the line device, the TSP is operational. LineInitializeEx initializes all lines of every configured TSP, not just the Cisco TSP. So, the number of *num line devices* is equal to all lines of every configured TSP.

Figure 6 TAPI Browser


TAPI32 Browser
File Options Help
Param: LAp+ LAp- Line+ Line- Call+ Call- PAp+ PAp- Pho+ Pho- Clear

```
lineGetDevCaps
lineGetDevCapsW
lineGetDevConfig
lineGetDevConfigW
lineGetIcon
lineGetIconW
lineGetID
lineGetIDW
lineGetLineDevStatus
lineGetLineDevStatusW
lineGetMessage
lineGetNewCalls
lineGetNumRings
lineGetProviderList
lineGetProviderListW
lineGetRequest
lineGetRequestW
lineGetStatusMessages
lineGetTranslateCaps
lineGetTranslateCapsW
lineHandoff
lineHandoffW
lineHold
lineInitialize
lineInitializeEx
```

```
LineApp=x800003ff
```

```
8:42.57.574 : Calling lineInitialize
8:42.57.684 : lineInitialize returned SUCCESS
  num line devs = 15
```

TAPI Phone Dialer

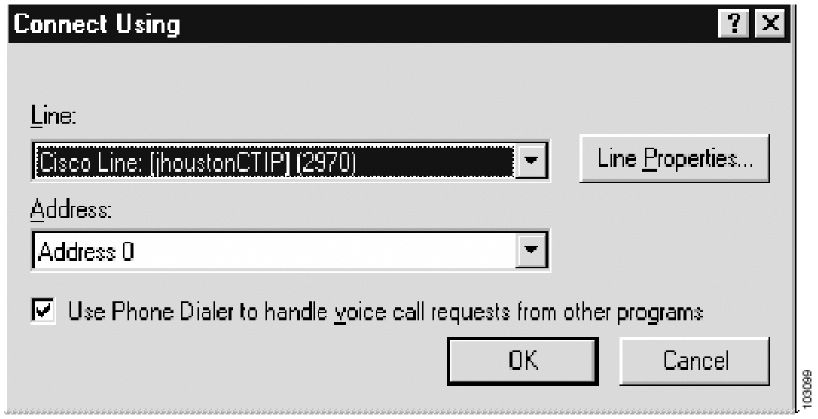
Another way to verify that the TSP is installed and configured properly is to use the TAPI Phone Dialer. The TAPI Phone Dialer is found on all of the Windows operating systems. You can do a Find to locate the file (dialer.exe).

The TAPI Phone Dialer is the same on Window 95/98/NT, but was changed in Windows 2000.

Windows 95/98/NT

For Windows 95/98/NT, start the **dialer.exe** program. If this is the first time Dialer has been used, the dialog box shown in Figure 7 is displayed. If not, you need to click *Tools* and *Select Connect Using*.

Figure 7 Windows 95/98/NT TAPI Phone Dialer


Connect Using

Line: Cisco Line: [jhoustonCTIP] (2970)

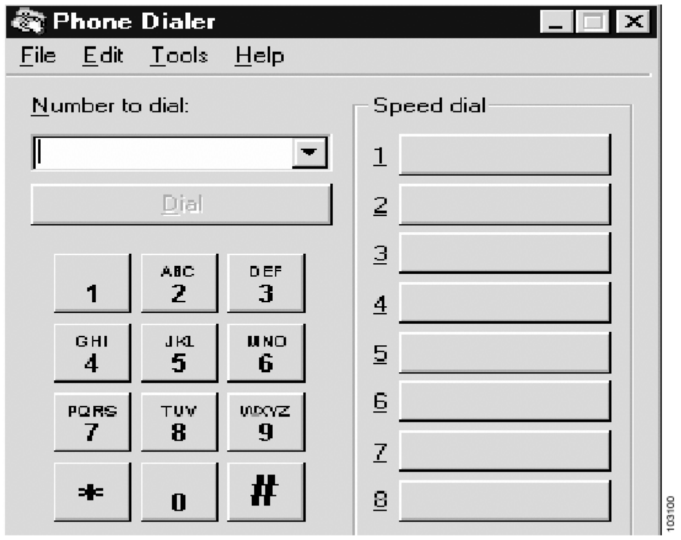
Address: Address 0

☑ Use Phone Dialer to handle voice call requests from other programs

In the *Connect Using* dialog box, select a *Cisco Line*. If there are no Cisco Lines in the line selection box, this indicates a connection problem between the Cisco TSP and the Cisco CallManager. Verify that all items in the "Postinstallation Checklist" on page 12 are correct. For the *Address* selection box, select Address 0. Click **OK** to continue.

Now type a number to dial. See Figure 8. If the phone call goes through, you know the Cisco TSP is operational.

Figure 8 Phone Dialer II

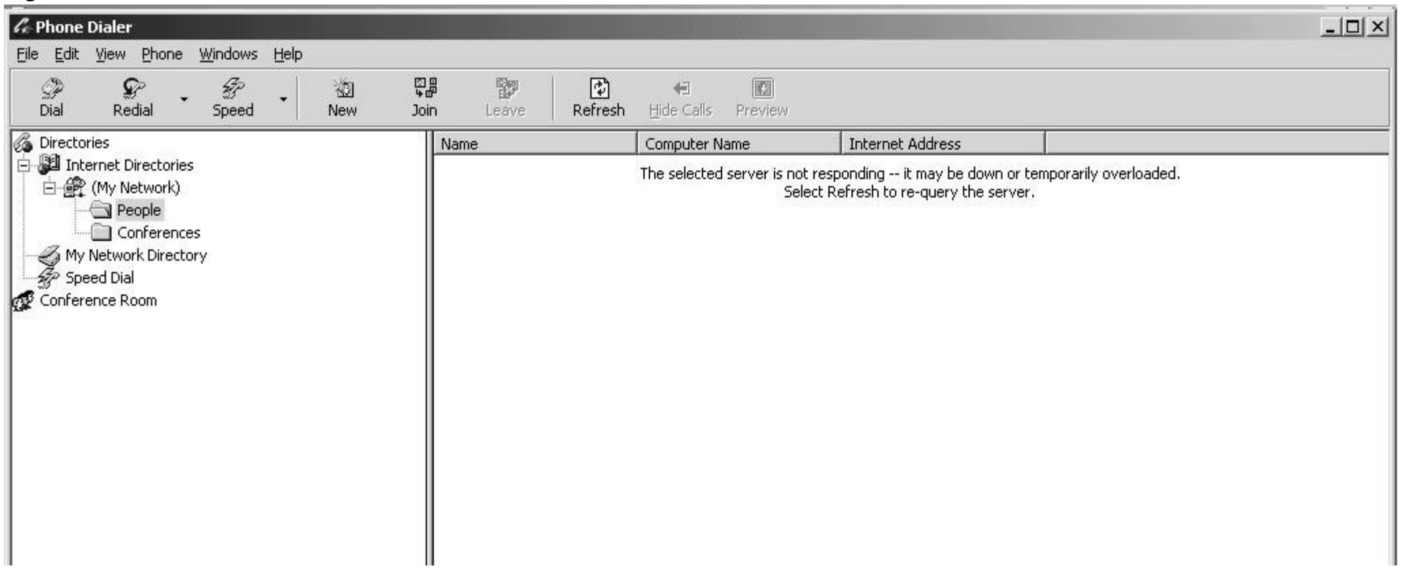


Windows 2000

The Phone Dialer for Windows 2000 is shown in Figure 9. It works like the Phone Dialer in Windows 95/98/NT, but is a little fancier because it is a TAPI 3.0 application. From the *Edit* menu, select *Options* to select the line you want to control.

Note The Cisco TSP is written as a TAPI 2.1 provider; however, in Windows 2000 and higher, Phone Dialer uses the TAPI 3.0 API layer. While TAPI 3.0 applications are not officially supported by the Cisco TSP, TAPI 3.0 versions of Phone Dialer should still work as described below for troubleshooting purposes.

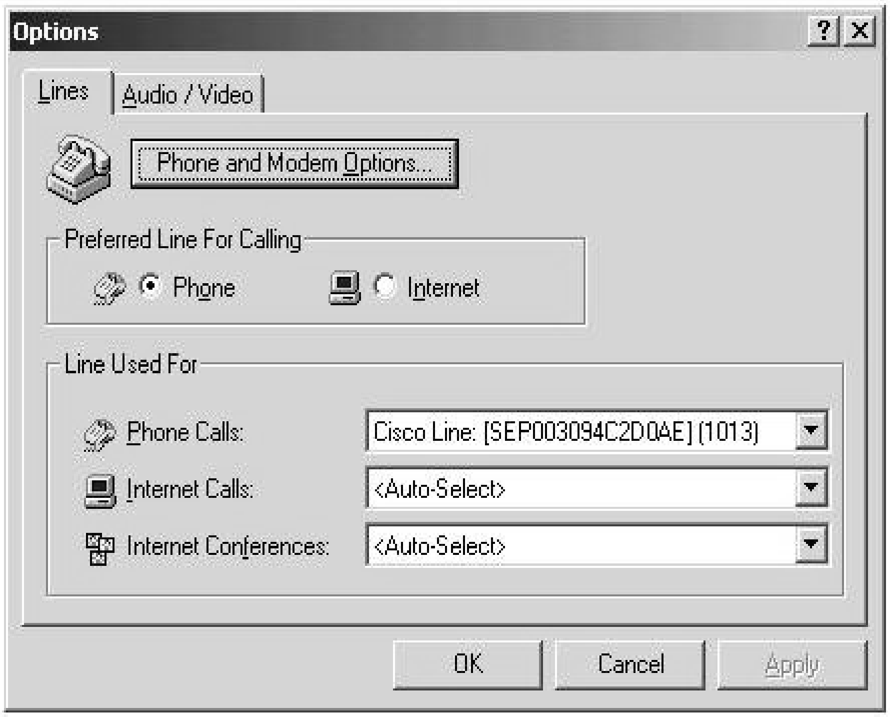
Figure 9 Windows 2000 TAPI Phone Dialer



The selected server is not responding -- it may be down or temporarily overloaded.
Select Refresh to re-query the server.

In the Lines tab, select *Phone* as the *Preferred Line For Calling*. See Figure 10. In the *Phone Calls* selection bod, select *Cisco Line*. If there are no Cisco Lines in the line selection box, this indicates a connection problem between the Cisco TSP and the Cisco CallManager. Verify that all items in the "Postinstallation Checklist" on page 12 are correct. Click **OK** for the settings to take effect.

Figure 10 Windows 2000 TAPI Phone Dialer II

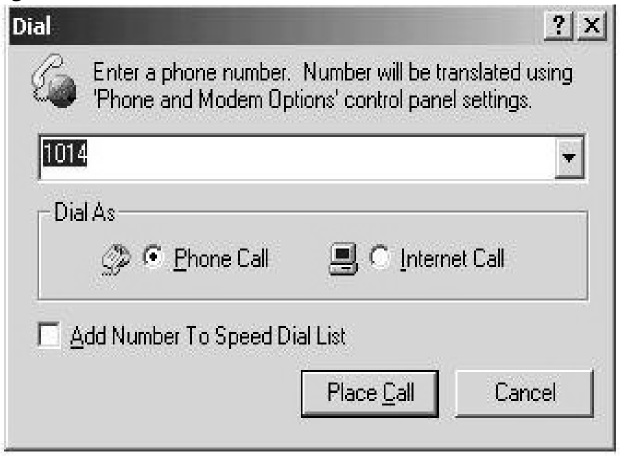


Preferred Line For Calling: [✓ Phone] Internet

Line Used For
Phone Calls: Cisco Line: [SEP003094C2D0AE] (1013)
Internet Calls: <Auto-Select>
Internet Conferences: <Auto-Select>

Next, click *Dial* and the dialog box in Figure 11 appears. Type the phone number you want to call and click *Place Call*. If the call goes through, the Cisco TSP is working properly.

Figure 11 Windows 2000 TAPI Phone Dialer III



Enter a phone number. Number will be translated using 'Phone and Modem Options' control panel settings.

1014

Dial As: [✓ Phone Call] Internet Call

☐ Add Number To Speed Dial List

Error Reporting

Tracing for the TAPI Service Provider

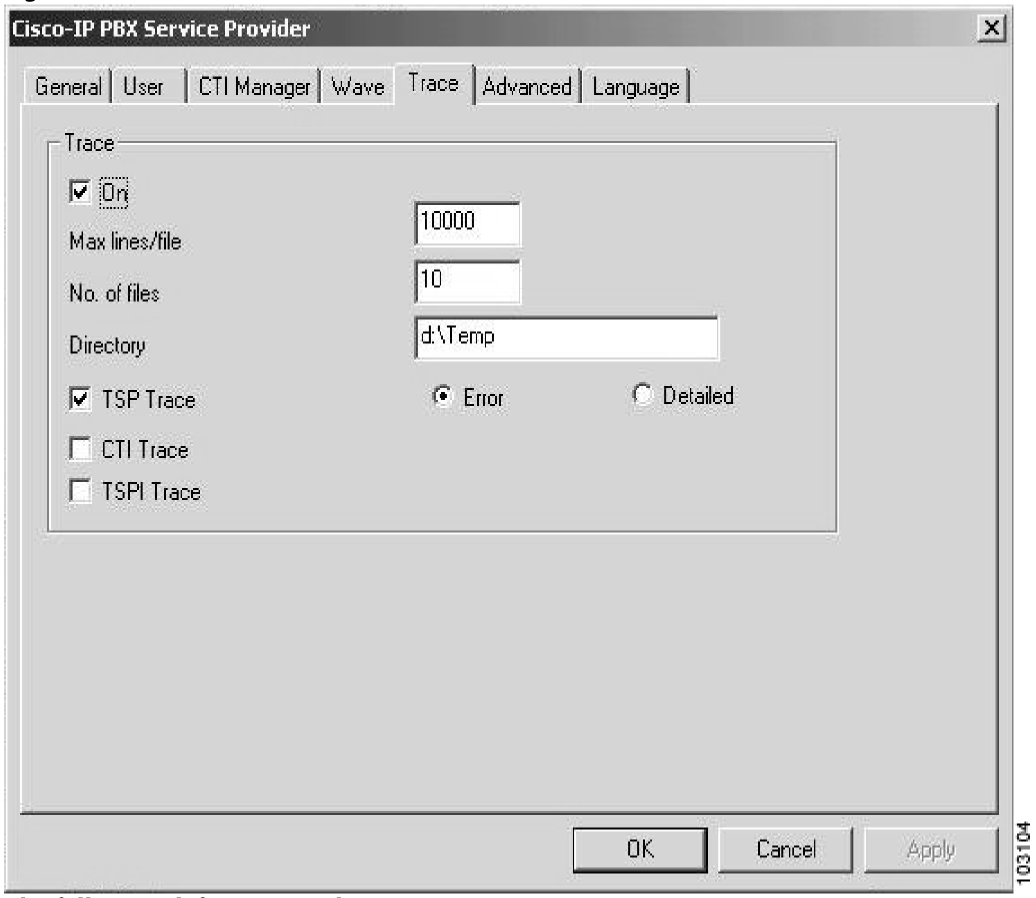
Perform the following steps to initiate tracing for the TAPI Service Provider:

1. Go to *Start->Settings->Control Panel* and select **Phone and Modem Options**.
2. Go to *Advanced* tab. Select **CiscoTSP0XX** and click **Configure**.
3. Go to *Trace* tab. Select **Trace On** check box (see Figure 12) and select the following:

Note It is recommended that all trace types be enabled when troubleshooting.

   a. **TSP Trace** to trace the TSP internal messages. Select **Error** to just log errors in the TSP. Select **Detailed** to log internal messages for debugging purposes.
   b. **CTI Trace** to trace the messages sent between CTI and TSP
   c. **TSPI Trace** to trace the requests and events sent between TSP and TAPI

Figure 12 TAPI Service Provider Trace



[✓] On
Max lines/file 10000
No. of files 10
Directory d:\Temp
[✓] TSP Trace [✓ Error] Detailed
[ ] CTI Trace
[ ] TSPI Trace

The following definitions apply:

Term	Definition
Directory	The path for the trace log. For example, c:\Temp.
No. of Files	Set this to a value greater than or equal to 1 enables rolling log files. For example, a value of 10 will cause up to 10 log files to be used in a cyclic fashion.
Max lines/files	Specifies the maximum number of trace statements that will be written to each log file. For example, a value of 1000 will cause up to 1000 trace statements to be written to each log file.

Error and Status Codes

TAPI error codes are reported within the TSP traces, as configured in “Tracing for the TAPI Service Provider” on page 17. Analysis of TSP trace logs is complex and error codes should be referred to the application developer support staff.

The following errors are normal operational warnings and can be safely ignored:

-
  ```
  CiscoTSP001.tsp| CSelsiusTSPLine::CallAsyncResponse() asyncResponseID=0x0000DCE1
  *ERROR* Did not   find call id 0x02044E99
  ```

This message is a normal message that results from an optimization in the TSP, whereby TSP execution is not blocked while waiting for a call tear-down confirmation from the CTI (this always succeeds). When the CTI does later return a message indicating successful tear-down of a closed call, the internal TSP call object may have already been destroyed, resulting in a “not found” call id.

-
  ```
  CiscoTSP001.tsp| CCtiInterface::ProcessResponse() *ERROR* Response does not map to an
  existing   request
  ```

  Another similar optimization as above. When an application closes a call, the TSP immediately generates failure replies for any outstanding asynchronous requests on the call. When the actual CTI responses for these pending requests arrive at the TSP later, their corresponding request objects may have already been destroyed and this line gets logged to the trace file.

-
  ```
  CiscoTSP001.tsp| CCtiInterface::RemoveFromSequenceNumberMap() *ERROR* Map entry does
  not exists   for sequenceNumber(0x0000DD33)
  ```

  Certain clean-up sequences in the TSP can result in attempting to remove a sequence map number from the internal request table more than once. This is a normal occurrence resulting from multi-threading issues and does not indicate a problem.

JTAPI Troubleshooting

This chapter contains the following topics:

Overview

Cisco JTAPI is a Java-based telephony applications programming interface that serves as a basic call control API.

JTAPI:

- Has primitive media support.
- Is full call and object oriented
- Provides application portability, i.e., independence from any particular telephony gear or any particular Operating System.

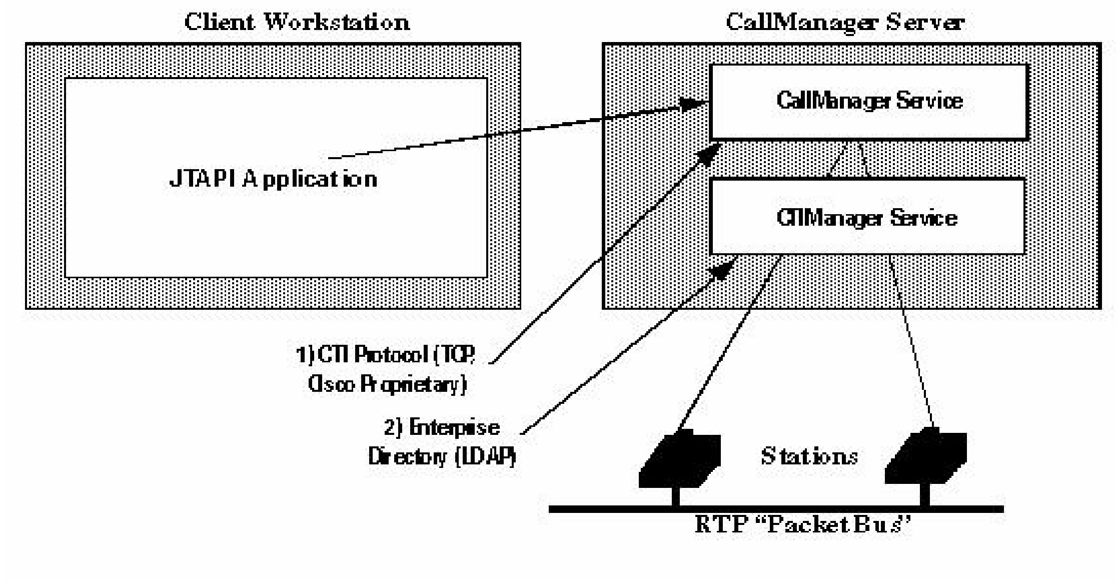
Note Cisco JTAPI is supported on Sun and Microsoft JVMs.

For more detailed information on JTAPI, refer to **http://java.sun.com/products/jtapi/**

Architecture

The Cisco JTAPI package abstracts the underlying Cisco CallManager architecture to applications. It communicates with the Cisco CallManager using a proprietary interface known as CTIQBE. See Figure 13.

Figure 13 JTAPI Architecture



Postinstallation Checklist

Some problems may be the result of improper installation. Use the following checklist to verify proper installation before proceeding with the troubleshooting process:

____ Check that the Cisco CTIManager location is configured properly in the application user interface.

____ Check to see if the Cisco CTIManager is up and running.

____ Check to see that the network connection is up between the JTAPI application server and Cisco CTIManager machines.

____ Check to see if the CallManager DC Directory Service is up and running.

____ Check that the Username and Password that is configured in the JTAPI application matches the Username and Password in the CallManager Directory.

____ Make sure that the Enable CTI Application Use flag has been checked in the Directory Administration on the CallManager for the TSP user.

____ Check that devices have been properly associated with the user in the CallManager Directory.

____ Check to see if the CTI Connection Limit has been reached.

____ Ensure the Cisco JTAPI version is compatible with the version of CallManager. The JTAPI version can be determined by running 'jview CiscoJtapiVersion' from the JTAPI server command line. This version can be verified by checking the CallManager Compatibility Matrix. Refer to the Defaul Loads section on the Cisco Connection Online website at http://www.cisco.com/univercd/cc/td/doc/product/voice/c_callmg/ccmcomp.htm#MinimumVersions

Troubleshooting Tools

Several sample applications are provided to verify the installation and to serve as sample code for developers. Sample applications can be found under the path *c:\Program Files\JTAPITools* and are named *makeCall* and *Jtrace*.

Note *Jtrace* and *makeCall* make use of the Microsoft Windows MFC and will not run under the Sun JDK.

From the command line, navigate to the makeCall directory and enter the following:

```
jview makecall <server name> <login> <password> <delay> <device1> <device2>
```

where:

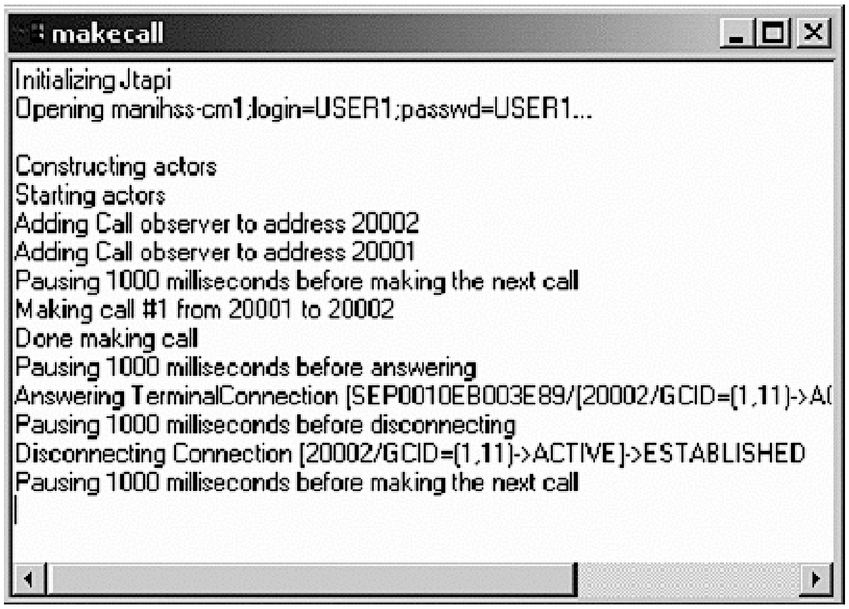
- *Server name* is the host name or IP address of the CTI Manager.
- *Login* and *password* are similar to those administered in the directory.
- Where *delay* is delay between two calls in milliseconds.
- *Device1* and *device2* are the directory numbers of the IP phones.

The application makes the call between 2 devices with an action delay of 1000ms until terminated. Figure 14 shows the result of a makeCall application that has been invoked with following parameters:

```
jview makecall manihss-cm1 USER1 USER1 1000 20001 20002
```

The application continues to make calls until the window is closed.

Figure 14 Running the makeCall application



```
Initializing Jtapi
Opening manihss-cm1;login=USER1;passwd=USER1...

Constructing actors
Starting actors
Adding Call observer to address 20002
Adding Call observer to address 20001
Pausing 1000 milliseconds before making the next call
Making call #1 from 20001 to 20002
Done making call
Pausing 1000 milliseconds before answering
Answering TerminalConnection [SEP0010EB003E89/[20002/GCID=(1,11)->A(
Pausing 1000 milliseconds before disconnecting
Disconnecting Connection [20002/GCID=(1,11)->ACTIVE]->ESTABLISHED
Pausing 1000 milliseconds before making the next call
```

From the command line, navigate to the jtrace directory and enter the following:

```
jview jtrace.Jtrace <server name> <login> <password> <device1> ... <deviceN>
```

where:

- Server name is the host name or IP address of the CTI Manager.

- Device1 and device2 are the directory numbers of the IP phones.
- Login and password are similar to those administered in the directory.
- Device1 ... deviceN are user controlled devices.

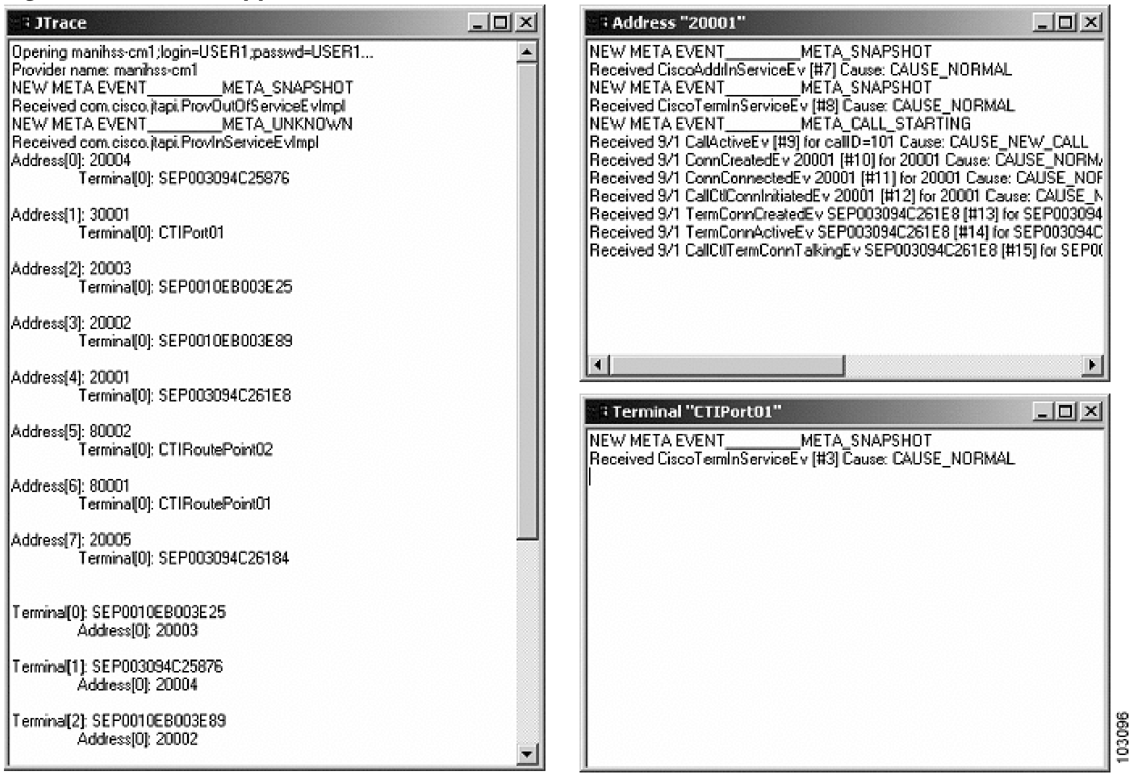
Figure 15 shows the results of a JTrace application that has been invoked with following parameters:

```
jview jtrace.Jtrace manihss-cm1 uesr1 user1 20001
```

The panel on the left shows the JTrace application window. The call manager, login, password, and a list of devices under this user control are listed. *Address* in JTAPI corresponds to a line or DN. *Terminal* corresponds to the device name.

The panel on the right top shows the JTAPI events when the device *20001* goes off hook. JTrace also opens a window for any CTI port in the application userid control list even if not explicitly specified.

Figure 15 JTrace Application



```
Opening manihss-cm1;login=USER1;passwd=USER1...
Provider name: manihss-cm1
NEW META EVENT__________META_SNAPSHOT
Received com.cisco.jtapi.ProvOutOfServiceEvImpl
NEW META EVENT__________META_UNKNOWN
Received com.cisco.jtapi.ProvInServiceEvImpl
Address[0]: 20004
          Terminal[0]: SEP003094C25876

Address[1]: 30001
          Terminal[0]: CTIPort01

Address[2]: 20003
          Terminal[0]: SEP0010EB003E25

Address[3]: 20002
          Terminal[0]: SEP0010EB003E89

Address[4]: 20001
          Terminal[0]: SEP003094C261E8

Address[5]: 80002
          Terminal[0]: CTIRoutePoint02

Address[6]: 80001
          Terminal[0]: CTIRoutePoint01

Address[7]: 20005
          Terminal[0]: SEP003094C26184


Terminal[0]: SEP0010EB003E25
          Address[0]: 20003

Terminal[1]: SEP003094C25876
          Address[0]: 20004

Terminal[2]: SEP0010EB003E89
          Address[0]: 20002
```

```
NEW META EVENT__________META_SNAPSHOT
Received CiscoAddrInServiceEv [#7] Cause: CAUSE_NORMAL
NEW META EVENT__________META_SNAPSHOT
Received CiscoTermInServiceEv [#8] Cause: CAUSE_NORMAL
NEW META EVENT__________META_CALL_STARTING
Received 9/1 CallActiveEv [#9] for callID=101 Cause: CAUSE_NEW_CALL
Received 9/1 ConnCreatedEv 20001 [#10] for 20001 Cause: CAUSE_NORM
Received 9/1 ConnConnectedEv 20001 [#11] for 20001 Cause: CAUSE_NOF
Received 9/1 CallCtlConnInitiatedEv 20001 [#12] for 20001 Cause: CAUSE_N
Received 9/1 TermConnCreatedEv SEP003094C261E8 [#13] for SEP003094
Received 9/1 TermConnActiveEv SEP003094C261E8 [#14] for SEP003094C
Received 9/1 CallCtlTermConnTalkingEv SEP003094C261E8 [#15] for SEP0
```

```
NEW META EVENT__________META_SNAPSHOT
Received CiscoTermInServiceEv [#3] Cause: CAUSE_NORMAL
```

Error Reporting

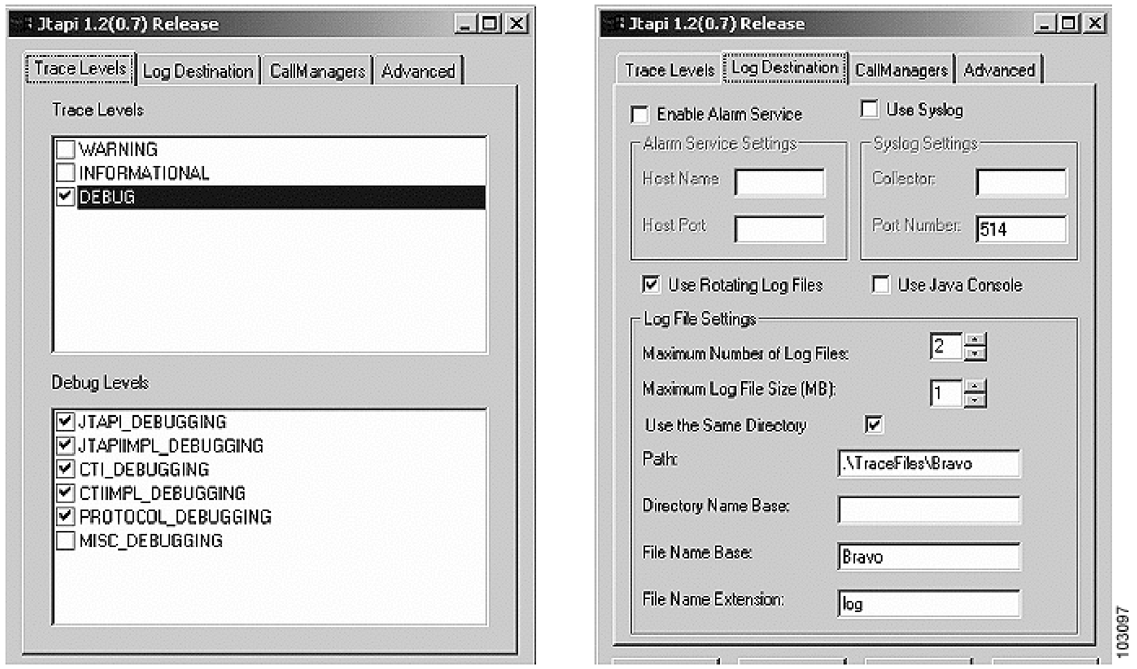
JTAPI Preferences Tool

The JTAPI Preferences Tool (JTPREFS) is a Windows-only tool that can be used to set tracing and error-logging levels and destinations, and to edit the default server names. See Figure 16.

Tracing and logging information can be sent to either rotating log files or to the system console. All settings are stored in the *JTAPI.INI* file.

Note Although the Cisco JTAPI implementation has been fully tested on Windows 2000 and Redhat Linux 7.2 platforms only, it is expected to function on other platforms, such as UNIX, if run under the supported Sun Microsystems JVM.

Figure 16 JTPREFS Tool



Trace Levels

☐ WARNING
☐ INFORMATIONAL
☑ DEBUG

Debug Levels

☑ JTAPI_DEBUGGING
☑ JTAPIIMPL_DEBUGGING
☑ CTI_DEBUGGING
☑ CTIIMPL_DEBUGGING
☑ PROTOCOL_DEBUGGING
☐ MISC_DEBUGGING

☐ Enable Alarm Service
☐ Use Syslog

Alarm Service Settings
Host Name:
Host Port:

Syslog Settings
Collector:
Port Number: 514

☑ Use Rotating Log Files
☐ Use Java Console

Log File Settings
Maximum Number of Log Files: 2
Maximum Log File Size (MB): 1
Use the Same Directory ☑
Path: .\TraceFiles\Bravo
Directory Name Base:
File Name Base: Bravo
File Name Extension: log

Error and Status Codes

For information on applicable error and status codes, refer to the JTAPI Developer Guide at http://www.cisco.com/univercd/cc/td/doc/product/voice/vpdd/cdd/3_3/3_3_3/jtapi/index.htm.

XML Services Troubleshooting

This chapter contains the following topics:

Overview

Extensible Markup Language (XML) services are applications for IP phones in Cisco CallManager environments. For example, an XML service can be used to display directory listings or to submit billing account numbers to be associated with calls. XML is a text-based markup language that is used to create documents that convey object information and the hierarchical structure of that information. XML services for IP phones can include the display of text or graphic information.

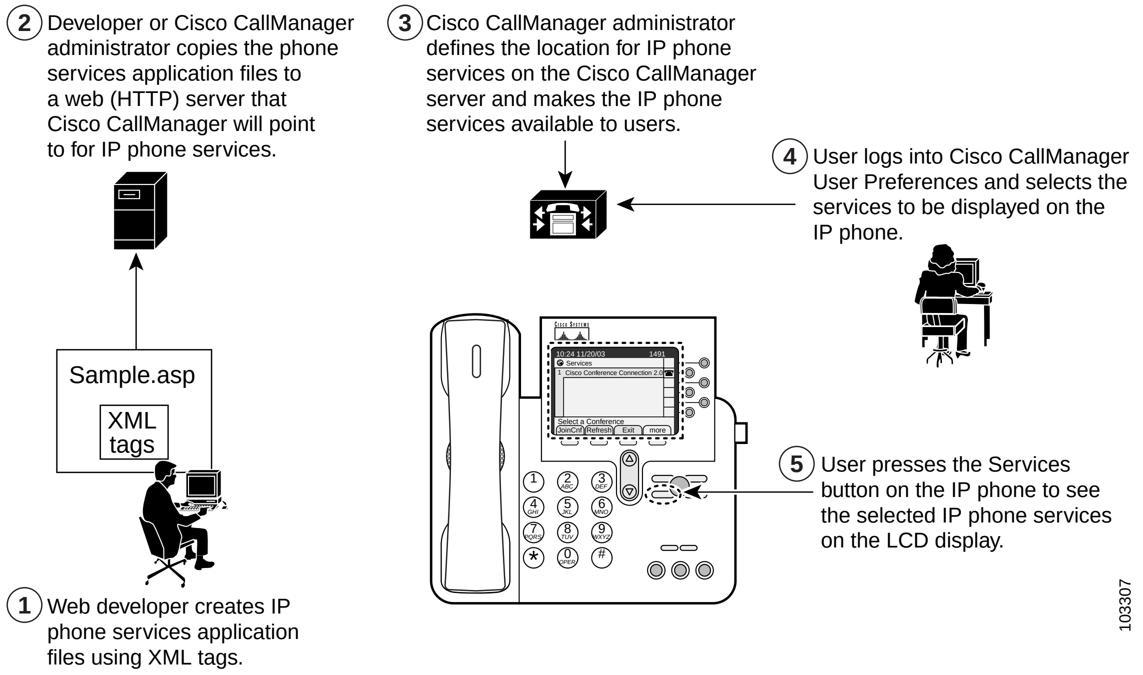
When an IP phone in a Cisco CallManager system uses a web-based XML service, it communicates with a web server using HTTP, the same protocol that PC web browsers use. Note that the applications themselves are written in XML and that Cisco IP phones do not understand HTML. For normal phone transactions, IP phones use other protocols to communicate. Messages between Cisco CallManager and an IP phone are sent using Skinny Client Control Protocol (SCCP). Voice packets are sent between phones connected in a call using Real-Time Transport Protocol (RTP), a UDP-based protocol especially designed for sending voice, video, and other time-sensitive data across packet networks.

An IP phone begins to invoke a service when the phone user presses the Services button. By default the Services button is assigned a URL that points to a web page called GetServicesMenu.asp on the Cisco CallManager server, although this default can be changed by using one of the Cisco CallManager administration screens.

The GetServicesMenu.asp web page displays a menu of phone services that have been specified for this phone. Each item on the service menu points to another URL on the web server where the XML application files for that service are stored. Figure 17 on page 28 illustrates the life cycle of an XML phone service application.

For more information, refer to the “Cisco IP Phone Services” chapter of the *Cisco CallManager System Guide* and the “Cisco IP Phone Services Configuration” chapter of the *Cisco CallManager Administration Guide.*

Figure 17 XML Services Life Cycle


2 Developer or Cisco CallManager administrator copies the phone services application files to a web (HTTP) server that Cisco CallManager will point to for IP phone services.
3 Cisco CallManager administrator defines the location for IP phone services on the Cisco CallManager server and makes the IP phone services available to users.
4 User logs into Cisco CallManager User Preferences and selects the services to be displayed on the IP phone.
Sample.asp
XML tags
5 User presses the Services button on the IP phone to see the selected IP phone services on the LCD display.
1 Web developer creates IP phone services application files using XML tags.
103307

Architecture

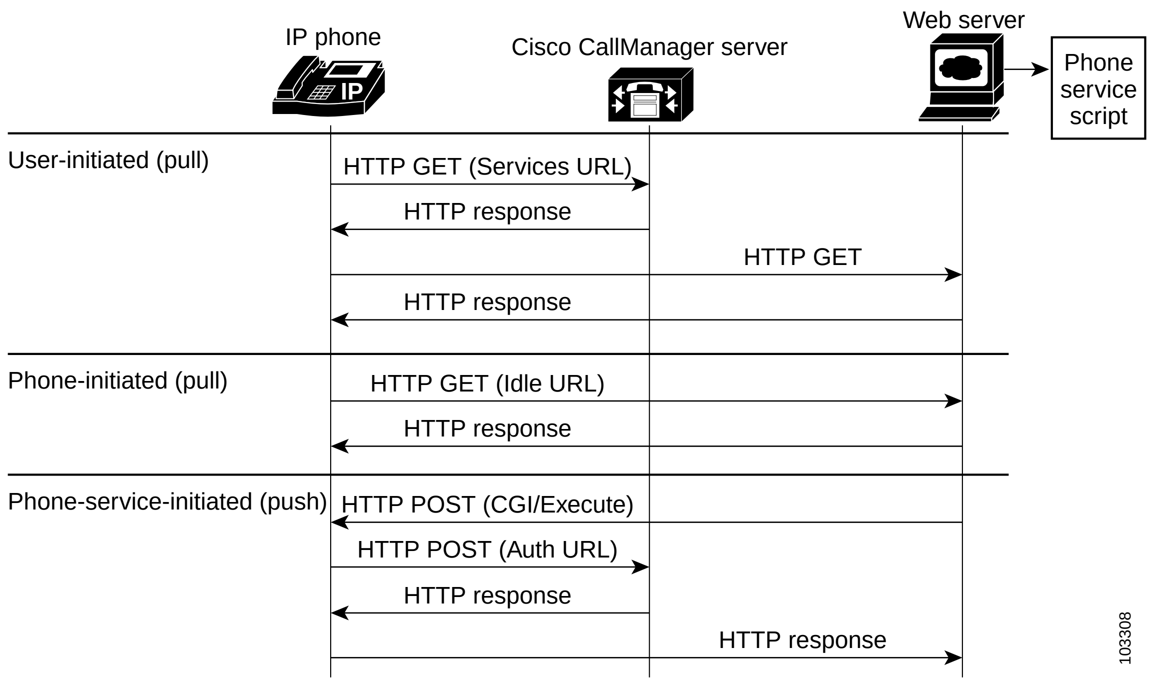
This section describes the following XML services concepts:

- XML Services Call Flow
- Architecture for Troubleshooting XML Services

XML Services Call Flow

HTTP call flows for XML services are shown in Figure 18 on page 29. Services can be initiated by the user, by the phone, or by another service that the phone is already using.

Figure 18 HTTP Web Services Request Call Flow



User-Triggered Service

When a user initiates a phone service, the following call flow is observed:

1. A user presses the Services button on an IP phone, and a menu of services is displayed.
2. The user selects a service and invokes it using a soft key on the phone.
3. The phone then sends an HTTP request to the URL on the web server that has been specified for the selected service.
4. The web server invokes the URL's ASP (Active Server Pages) or JSP (Java Server Pages) service.
5. The web server processes information and returns an XML response to the phone.
6. The phone uses an embedded web server to properly display the XML response to the user.

Event-Triggered Service

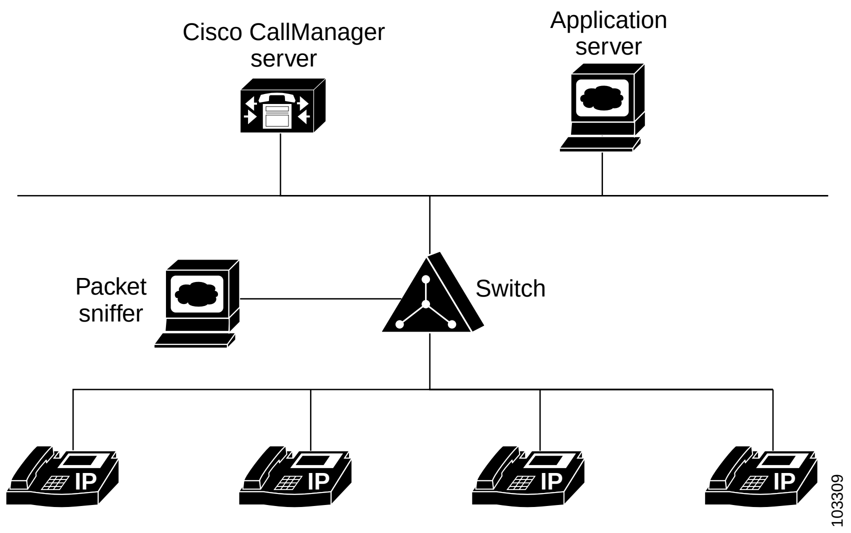
When an event initiates a phone service, the following call flow is observed:

1. An event is triggered through call processing or some other scenario that triggers a service.
2. Call information is transferred to the service from an application that is monitoring calls.
3. The service processes the information, formats the response, and locates the phone to which to push the data.
4. The XML data is pushed to the phone and presented to the user.

Architecture for Troubleshooting XML Services

Figure 19 shows the suggested architecture for troubleshooting XML services on Cisco CallManager networks, in which a packet sniffer is located at the switch. Typically, you collect sniffer traces by connecting a laptop or other sniffer-equipped device to a Catalyst port that is configured to span the VLAN or specific port(s), directly connected at the router, that contain the trouble information. If no free port is available, connect the sniffer-equipped device to a hub that is inserted between the switch and the device.

Figure 19 Architecture for Troubleshooting XML Services



Troubleshooting Tools

The following tools will help you troubleshoot problems with XML services.

- Standard web browser (Microsoft Internet Explorer 5.0 or a later version)

  Verifies authentication or connection issues to and from the XML services URL and the phone.
- Text editor such as Microsoft Notepad or an HTML editor

  Allows you to analyze the syntax of the XML code and make changes where necessary.
- Network packet analyzer such as Ethereal or SnifferPro

  Verifies exactly what data is being sent among the Cisco CallManager, the application server, and the phones.

- Validator

  Parses the XML content of a file or web page and authenticates it against a schema for Cisco IP phone XML objects. The XML Validator is a command-line tool that is available on a CD-ROM in the Cisco Press book called *Developing Cisco IP Phone Services*. Instructions for XML Validator can be found in Chapter 11 of that book. The following is an example of a validated file from Validator:

```
<?xml version="1.0"?>
<CiscoIPPhoneMenu>
    <Title>Cisco Phones Are Great, Right?</Title>
    <Prompt>What do you want to do today?</Prompt>
    <MenuItem>
        <Name>Yahoo</Name>
        <URL>http://www.yahoo.com/index.html</URL>
    </MenuItem>
</CiscoIPPhoneMenu>
```

  The following is an example of a validation error reported by Validator:

```
MS Parser reported this error:
Code = 0xc00ce014
Source = Line : b Char : 11
Error Description = Element content is invalid according to the DTD/Schema.
Expecting: Input Flags.

Source text where the error occurred:

              <DefaultValue></DefaultValue>
--------------^
```

Troubleshooting Checklist

The following tips apply to troubleshooting Cisco IP phone service applications:

- Microsoft Internet Explorer 5 or higher can display the XML source with its default style sheet.
- Standard IP troubleshooting techniques are important for diagnosing HTTP errors.
- Understanding the syntax and programming techniques for XML development can help you to diagnose problems.

Use the following checklist for initial troubleshooting of common problems.

____ Ensure that the user is associated with the device. The username and password that are passed to Cisco CallManager for authentication must have a corresponding user who is associated with the device that will receive the PUSH. See “PUSH-Related Issues” on page 32.

____ Check for parsing issues. See “Parsing Issues and Syntax Errors” on page 33.

____ If the AXL client application is unable to connect to the AXL service, check the following:

____ Ensure that the web service is reachable from the Cisco CallManager and the IP phone. From your web browser, type the URL of the web service page that has been configured in Cisco CallManager to confirm that the service is reachable. You can perform a quick check at the phone by pressing the Services button and selecting the service that has been configured.

____ Externally verify the name resolution for URLs and whether the phone has DNS set. If DNS is suspected as a source of trouble, use IP addresses in URLs.

____ Troubleshoot the PUSH-related issue. See “PUSH-Related Issues” on page 32.

If you still cannot locate the root cause for your problem, examine some packets using the Packet Sniffer. Typically, you collect sniffer traces by connecting a laptop or other sniffer-equipped device to a Catalyst port that is configured to span the VLAN or specific port(s), directly connected at the router, that contains the trouble information. If no free port is available, connect the sniffer-equipped device to a hub that is inserted between the switch and the device.

To use the Packet Sniffer, perform the following steps:

Step 1 Connect the Packet Sniffer to the switch.

Step 2 Set the Packet Sniffer to "span ports" and turn all filtering off.

Step 3 Turn on the Packet Sniffer.

Step 4 Try running your service again normally.

Step 5 Once the error condition has occurred, stop the Sniffer capture and save the file.

Step 6 Report the problem to Cisco using the process described in "Error Reporting" on page 33.

When reporting a problem, be sure to have available the IP and MAC addresses of all equipment that is involved, such as IP phones, gateways, and Cisco CallManager servers.

Service Connection

Copy the service URL of your service as configured in Cisco CallManager and paste it into the web browser on your PC. Then analyze the response to determine whether the service can be reached.

Use a logged Telnet session to verify that the desired HTTP headers are returned. Telnet to the server on port 80; then, enter get /*path*/*page*.

PUSH-Related Issues

A PULL results from a request made at the phone, whereas a PUSH results from the web service initiating content to be sent to the phone. When troubleshooting PUSH problems, use the following simple HTML page to verify that the username and password are correct:

```
<HTML>
  <HEAD>
    </HEAD>
      <BODY>
         <FORM action="http://10.0.0.1/CGI/Execute" Method="POST">
         <TEXTAREA NAME="XML" Rows="20" Cols="80">
              <CiscoIPPhoneExecute>
              <ExecuteItem Priority="0" URL="Play:chime.raw"/>
              </CiscoIPPhoneExecute>
         </TEXTAREA>
          <BR>
          <input type=submit value=POST>
        </FORM>
      </BODY>
</HTML>
```

This HTML page provides a text area for entering the XML object to be PUSHed to the phone. You need to replace 10.0.0.1 in the example above with the IP address of your phone. When the POST button is clicked, the object will be PUSHed to the phone. The phone will then force the browser to authenticate, and a username and password window will open. After authenticating, the phone will return the <CiscoIPPhoneResponse> object, which will be viewable in the web browser if you are using an XML-capable browser, such as Internet Explorer 5 or a later version. Using this simple HTML page allows you to quickly rule out any authentication or XML parsing and validation issues with your application.

Parsing Issues and Syntax Errors

The XML Validator can be used to validate the syntax of the XML code as a first step.

When troubleshooting XML parsing errors, do not exceed the maximum field lengths allowed by the XML schema. For example, the <Title> and <Prompt> fields are each limited to 32 characters (31 characters in some older firmware loads). A good reference for XML syntax and restrictions can be found in the Cisco Press book *Developing Cisco IP Phone Services*. For more detailed information, refer to the Cisco IP Phone XML Schema file (xsd) that is included in SDK version 3.3(4) or later. This XML-standard schema file defines all of the details of the allowed XML objects and can be used with XML editing software packages to greatly simplify the process of creating and validating Cisco IP phone XML objects.

The following tips apply to troubleshooting XML parsing errors in Cisco IP phone services applications:

- Verify the object tags (the object tags are case-sensitive).
- Verify that the ampersand (&) and the other four special characters in XML are used according to their restrictions while inside XML objects. By XML convention, the XML parser also requires that you use an escape sequence for five special characters, which are listed in Table 2.

Table 2 Escape Sequences for Special Characters

Name	Character	Escape Sequence
Ampersand	&	&amp;
Apostrophe	'	&apos;
Left angle bracket	<	&lt;
Right angle bracket	>	&gt;
Quote	"	&quot;

Error Reporting

When reporting problems with IP phone services (IPPS), be prepared to provide the following information:

- Sniffer traces connected at the switch and at the back of the phone.
- A detailed description of the network elements involved and their respective IP addresses in sniffer traces.
- A brief description of the function of the service and the problem found.
- The XML source code that shows the XML request that was sent to the phone and the XML response from the phone that was received by the application.
- The code used in the service application. This code is also helpful when you are trying to determine the root cause of problems.

Error and Status Codes

An error or status code is received by the XML application as an XML response from the web browser that is resident in the phone.

An XML error tag appears in the following format:

```
<CiscoIPPhoneError Number="x"/>
```

The following error codes can result from a PUSH.

- Error 1 = Error parsing CiscoIPPhoneExecute object
- Error 2 = Error framing CiscoIPPhoneResponse object
- Error 3 = Internal file error
- Error 4 = Authentication error

The following error messages may appear on the prompt line of the Cisco IP phone display.

- XML Error [4] = XML Parser Error (Invalid Object)
- HTTP Error [8] = Unknown HTTP Error
- HTTP Error [10] = HTTP Connection Failed

AXL Troubleshooting

This chapter contains the following topics:

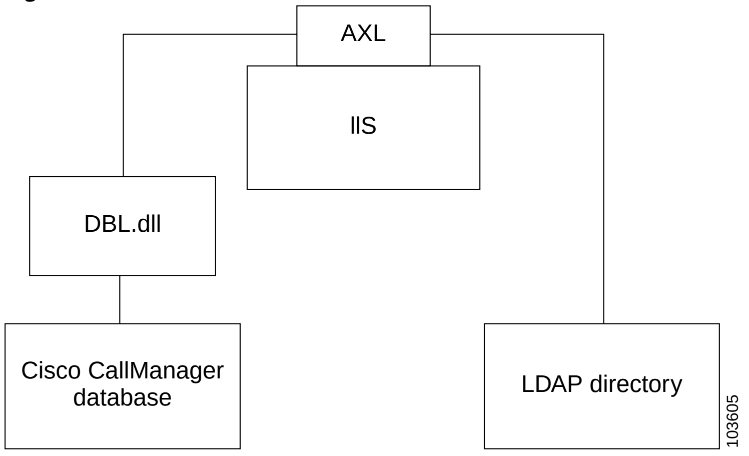
Overview

AVVID XML Layer (AXL) is a Cisco application programming interface (API) and web service designed to give applications access to Cisco CallManager configuration and provisioning services. AXL is implemented as a Simple Object Access Protocol (SOAP) over HTTP web service, in which requests in the form of extesnible markup language (XML) documents are sent from the application to the Cisco CallManager's web server, which responds with an XML-formatted response. Security is handled through the Cisco CallManager's Internet Information Services (IIS) security mechanism.

Architecture

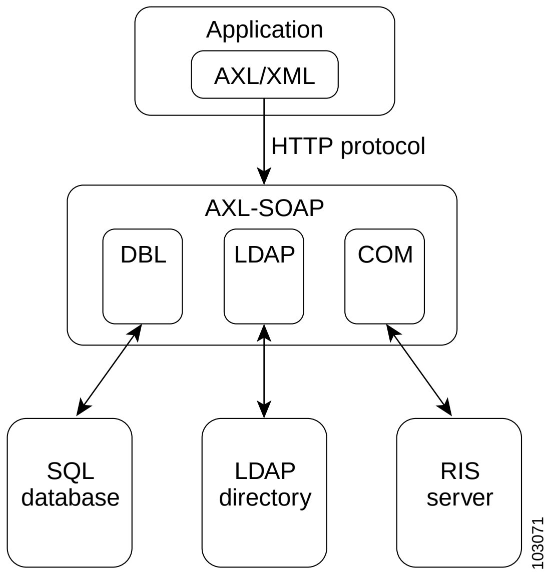
AXL is hosted as an IIS dynamic-link library or ISAPI.dll (SOAPISAPI.dll) as shown in Figure 20. AXL accesses either the Cisco CallManager configuration database, or the embedded DC-Directory Lightweight Directory Access Protocol (LDAP) directory to read or write configuration data. Database access is handled by the DBL.dll layer, which enforces the Cisco CallManager database business rules. This same DBL.dll layer serves the Cisco CallManager Administration program’s graphical user interface (GUI) pages.

Figure 20 AXL Architecture



The AXL API methods, known as requests, use a combination of HTTP and SOAP as shown in Figure 21. SOAP is an XML remote procedure call protocol. Users perform requests by sending XML data to the Cisco CallManager Real-Time Information Server (RIS). The server then returns the AXL response, which is also a SOAP message.

Figure 21 AXL Request Flow



Postinstallation Checklist

The AXL web service is enabled by default on Cisco CallManager servers and does not have to be installed or configured. Use the following checklist to avoid some common problems that may be avoided by fine-tuning your configuration before proceeding with the troubleshooting process:

____ If the AXL client application is unable to connect to the AXL service, check the following:

- The AXL application has been configured with the correct IP address of the AXL server.

- The AXL application has been configured with the appropriate AXL user credentials.
- The user credentials have the correct access rights, as described in the “Postinstallation Checklist” on page 36.
- The application server has HTTP connectivity to the AXL server (that is, port 80).
- HTTPS (secure) has been configured for AXL.

____ If the AXL functions or requests are failing, check the following:

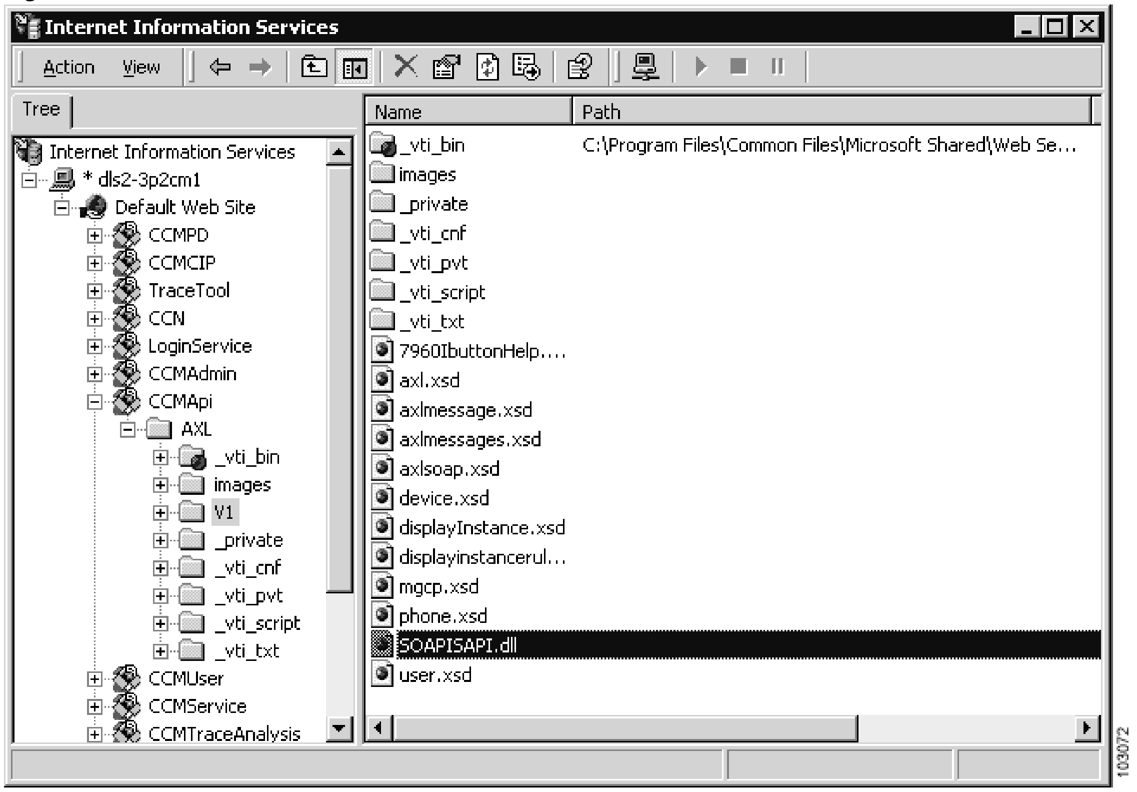
- AXL logs for AXL or SOAP error responses. See “Error Codes” on page 43.
- IIS logs for general web server errors or problems

Note DBL logs should be enabled only on request from Cisco TAC or Cisco Developer Services.

____ Check that users other than the Microsoft Windows Administrator group that require access to AXL have read and execute rights to the SOAPISAPI.dll file.

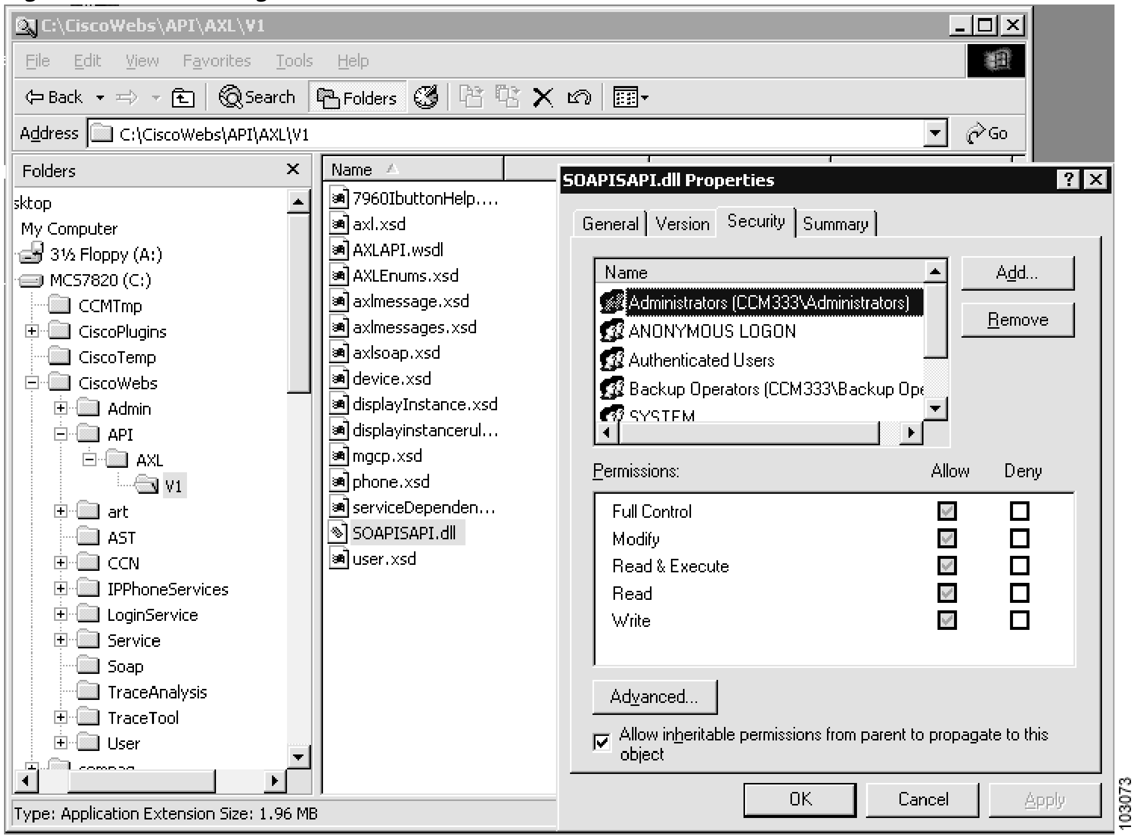
By default, only the Windows Administrator group is given access to the SOAPISAPI.dll file, restricting AXL access to the administrator. Figure 22 shows the location of the SOAPISAPI.dll file.

Figure 22 IIS SOAPISAPI.dll File



To enable AXL access to additional users, simply give the new users read and execute rights to the file C:\CiscoWebs\API\AXL\V1\SOAPISAPI.dll. Figure 23 shows the Properties window through which you can add and change access rights to a file.

Figure 23 Access Rights for SOAPISAPI.dll File



Name:
- Administrators (CCM333\Administrators)
- ANONYMOUS LOGON
- Authenticated Users
- Backup Operators (CCM333\Backup Ope
- SYSTEM

Permissions:	Allow	Deny
Full Control	✓	☐
Modify	✓	☐
Read & Execute	✓	☐
Read	✓	☐
Write	✓	☐

☑ Allow inheritable permissions from parent to propagate to this object

____ Check that applications in a cluster configuration are connected to the AXL service only on the Cisco CallManager Publisher server.

____ Verify basic AXL functionality by performing the procedure that follows:

Verifying AXL Functionality Procedure

Step 1 Open Notepad on the client PC by choosing **Start** > **Programs** > **Accessories** > **Notepad**.

Step 2 Copy the following HTML text into the untitled file:

```
<html>
<body>
<form method="POST" action="http://**CCM IP**/CCMAPI/AXL/V1/SOAPISAPI.dll">
  <input type="text" name="XML" size="20"><input type="submit">
</form>
</body>
</html>
```

Step 3 Change **CCM IP ** to the IP address of the AXL Cisco CallManager server.

Step 4 Save the file to the desktop as axltest.htm.

Step 5 Open the file with Internet Explorer.

Step 6 Click the **Submit Query** button.

Step 7 Enter the AXL access user credentials.

Step 8 Verify that the resulting page is similar to the following:

```
- <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>The AXL API service can only accept Content-Type: text/xml</faultstring>
  </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

The resulting page confirms that the client PC has HTTP connectivity to the AXL server, that the AXL user credentials are valid, and that the AXL service is running.

____ Check the Cisco Database Layer service parameter MaxAXLWritesPerMinute:

Updating the Cisco CallManager database can have the side effect of adversely affecting system performance. To prevent this effect, the system administrator can control how many AXL requests are allowed to update the database per minute through the Database Layer service parameter MaxAXLWritesPerMinute.

AXL accommodates all requests until the MaxAXLWritesPerMinute value is reached. The default value of this parameter is 20, and the maximum configurable value is 999. However, the maximum Cisco supported value for production systems is 60.

After the MaxAXLWritesPerMinute value is reached, 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 MaxAXLWritesPerMinute value is reached.

Troubleshooting Tools

This section describes available troubleshooting tools.

AXL Trace Logs

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

AXL trace logs are created in the C:\Program Files\Cisco\Trace\AXL directory. AXL creates up to 50 1-MB text files in this directory. Once 50 files have been written, the oldest file is overwritten by the newest file.

Note Restarting the Cisco CallManager's WWW service (IIS), either manually or with a machine reset, causes trace logs to begin overwriting the oldest file.

The AXL trace log contains:

- Time that the request was received
- Client IP address
- Client user ID
- Request ID number

- Request contents
- Response contents

The following is sample AXL trace log output:

```
RequestTime: 01/15/2003-11:24:49
Client IP: 10.101.156.108
Client User ID: user1
RequestID: 1
RequestContents:
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <SOAP-ENV:Body> <m:getPhone
xmlns:m="http://www.cisco.com/AXL/API/1.0" sequence="1"> <phoneName>Joe_Phone</phoneName>
<!-- <phoneId>{2FC4F3C0-FC99-4646-A44A-137356B6289C}</phoneId>  --> </m:getPhone>
</SOAP-ENV:Body> </SOAP-ENV:Envelope>
ResponseContents:
<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[Item not
valid: The specified phone was not found.]]></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://user1-w2k/CCMApi/AXL/V1/axlsoap.xsd">
<axl:error  sequence="1"> <code>2</code> <message> <![CDATA[Item not valid: The specified
phone was not found.]]></message> <request>getPhone</request> </axl:error> </detail>
</SOAP-ENV:Fault> </SOAP-ENV:Body> </SOAP-ENV:Envelope>
```

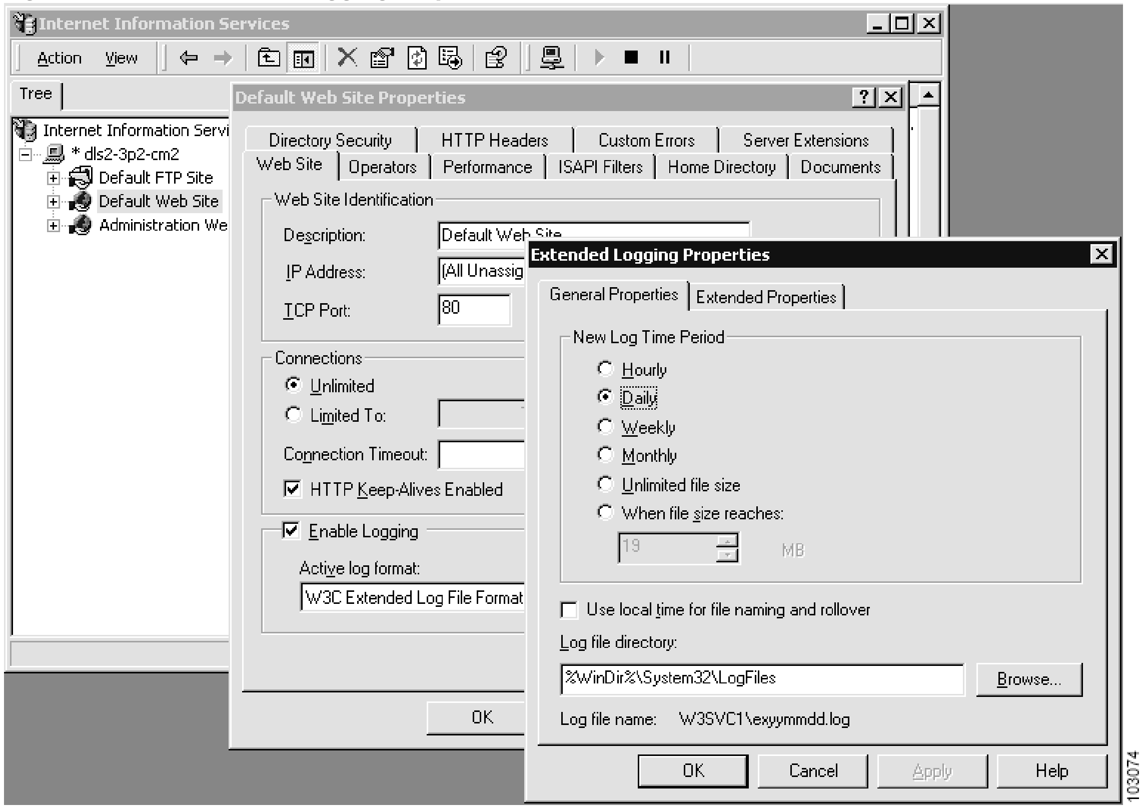
IIS Logs

IIS logs record user sessions with the IIS web service. Information recorded is:

- Username
- IP address
- Access method
- Uniform resource identifier (URI) requested
- Potentially other detailed information about the session and client

IIS logs are configured from the IIS manager. Only a few of the possible IIS logging information fields are enabled by default. Refer to the Microsoft IIS documentation for more information about the additional logging options shown in Figure 24.

Figure 24 IIS Extended Logging Properties



Web Site Identification
Description: Default Web Site
IP Address: (All Unassig
TCP Port: 80

Connections
[✓ Unlimited]
Limited To:
Connection Timeout:
[✓] HTTP Keep-Alives Enabled

[✓] Enable Logging
Active log format: W3C Extended Log File Format

New Log Time Period
Hourly
[✓ Daily]
Weekly
Monthly
Unlimited file size
When file size reaches: 19 MB

[ ] Use local time for file naming and rollover
Log file directory: %WinDir%\System32\LogFiles
Log file name: W3SVC1\exyymmdd.log

Note IIS serves many functions on Cisco CallManager. Not all IIS log entries are related to AXL access.

DBL Logs

Database Layer (DBL) logs document the interaction between the AXL web service and the supporting DBL and Structured Query Language (SQL) database services. Typically these logs are necessary only when debugging a problem internal to AXL, DBL, or SQL.

Perform the following steps to enable DBL logs:

Step 1 From the Cisco CallManager Administration window, choose **Application > Cisco CallManager Serviceability**.

The Cisco CallManager Serviceability window shown in Figure 25 displays.

Step 2 Choose **Trace > Configuration**.

Step 3 From the Servers column, choose the server.

The server that you chose displays next to the Current Server title, and a box with configured services displays.

Step 4 From the Configured Services box, choose the **Cisco Database Layer Monitor** service.

The service that you chose displays next to the Current Service title, along with the current server that you chose. The trace parameters display for the service that you chose.

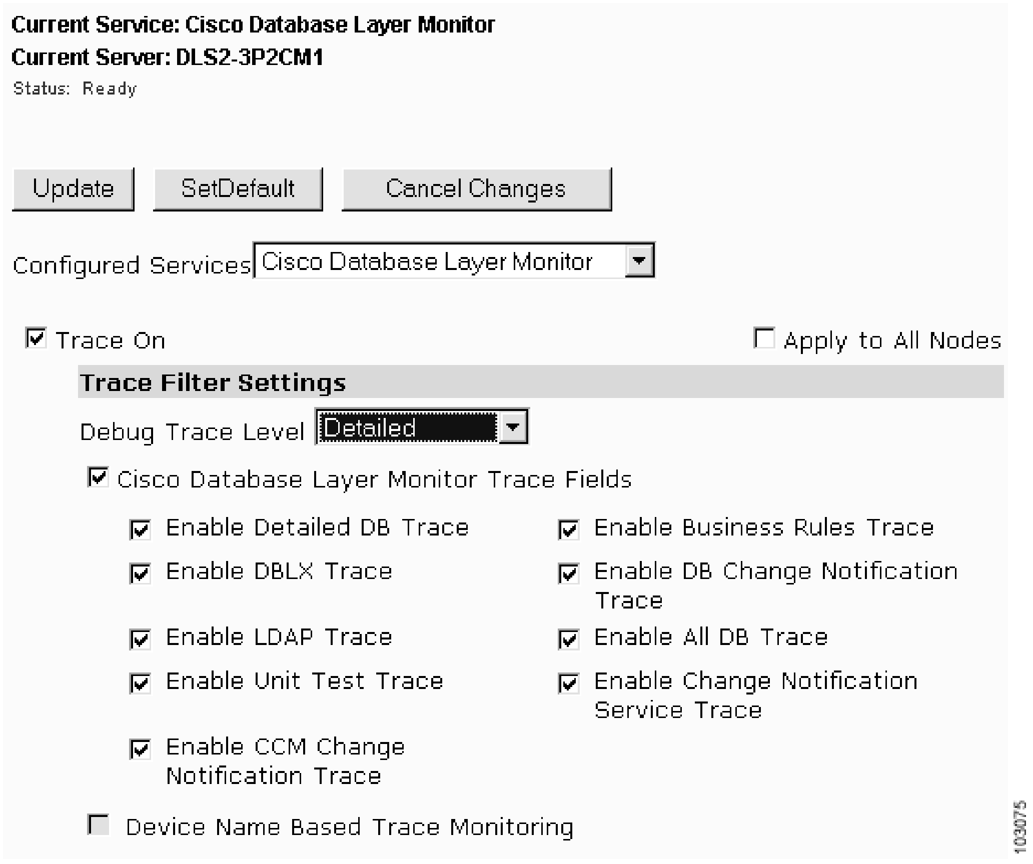
Note Only the trace parameters for the service that you chose display. The display shows all other parameters dimmed.

Step 5 Check the Trace On check box.

Step 6 If you want trace to apply to all Cisco CallManager servers in the cluster, check the Apply to All Nodes check box.

Step 7 In the Debug Trace Level field, click the drop-down arrow and click **Detailed**.

Figure 25 Cisco CallManager Serviceability Window



Current Service: Cisco Database Layer Monitor
Current Server: DLS2-3P2CM1
Status: Ready

Configured Services: Cisco Database Layer Monitor

[✓] Trace On [ ] Apply to All Nodes

Trace Filter Settings

Debug Trace Level: Detailed

[✓] Cisco Database Layer Monitor Trace Fields

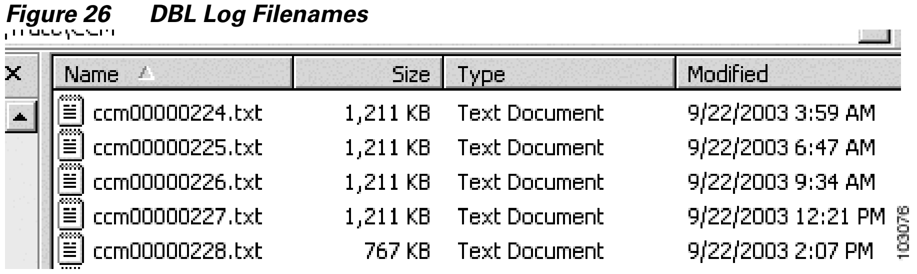
[✓] Enable Detailed DB Trace	[✓] Enable Business Rules Trace
[✓] Enable DBLX Trace	[✓] Enable DB Change Notification Trace
[✓] Enable LDAP Trace	[✓] Enable All DB Trace
[✓] Enable Unit Test Trace	[✓] Enable Change Notification Service Trace
[✓] Enable CCM Change Notification Trace	

[ ] Device Name Based Trace Monitoring

Note the output trace file path for later log collection.

The DBL Layer tracing mechanism uses circular logging, which means that log entries are written to a single log file until it reaches a certain configurable size. Once that size has been reached, subsequent log entries are written to a new log file with a sequentially incremented filename, as shown in Figure 26.

Figure 26 DBL Log Filenames



Name	Size	Type	Modified
ccm00000224.txt	1,211 KB	Text Document	9/22/2003 3:59 AM
ccm00000225.txt	1,211 KB	Text Document	9/22/2003 6:47 AM
ccm00000226.txt	1,211 KB	Text Document	9/22/2003 9:34 AM
ccm00000227.txt	1,211 KB	Text Document	9/22/2003 12:21 PM
ccm00000228.txt	767 KB	Text Document	9/22/2003 2:07 PM

The number of log files available is also a limited configurable number. When the last log file is full, the logging process deletes the first log file and reuses its filename. File overwriting proceeds in a circular manner, with newer log files overwriting existing log files with the same name.

When you work with circular logging systems, it is important that you balance the disk space requirement, which is calculated by multiplying the maximum size of a log file by the maximum number of log files allowed, with the required log file collection interval. If the size and number of log files are too large, a large amount of disk space can be consumed by logging. If the size and number of log files are too small, log files can be overwritten before the administrator can collect them.

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, such as in the following example:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Client</faultcode>
            <faultstring>
            <![CDATA[
            An error occurred during parsing
            Message: End element was missing the character '>'.

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

SOAP Fault messages can also contain more detailed information. The following 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/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 will be 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 subelements 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 described in Table 3.

Table 3 Error Code Descriptions

Error Code	Description
Less than 5000	These are errors that directly correspond to DBL Exception error codes.
5000	Unknown Error—An unknown error occurred while processing the request. This error can be caused by a problem on the server, but it 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. 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, this error is returned if the server expects the next node to be <name>, but encounters <protocol>. 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.

SCCP (Skinny) Phone/Endpoint Troubleshooting

This chapter contains the following topics:

Overview

The Skinny Client Control Protocol (SCCP) comprises of a messaging set between a skinny client and the Cisco CallManager call-processing server and is a lightweight alternative to H.323. All of the Cisco IP Phones communicate with the Cisco CallManager through SCCP.

A skinny client uses:

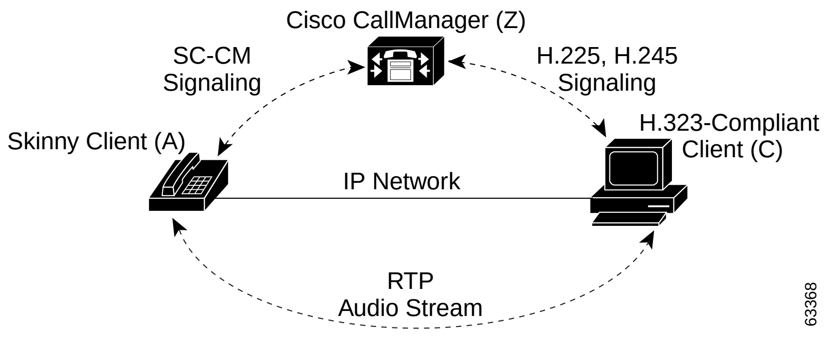
- TCP/IP to and from one or more Cisco CallManagers to transmit and receive stimulus.
- RTP/UDP/IP to and from a similar skinny client or H.323 terminal for audio.

SCCP is a stimulus-based protocol and is designed as a communications protocol for hardware endpoints and other embedded systems, with significant CPU and memory constraints.

Architecture

Figure 27 shows the interaction between skinny and H.323 clients through the Cisco CallManager server. The skinny client signaling is from the phone to the Cisco CallManager server. The Cisco CallManager server in turn translates the call-control messages to the H.323 client using H.225 for call routing and H.245 for negotiation. After the call is established, the RTP audio stream runs directly between both clients.

Figure 27 Skinny Client to/from H.323 Client Example



Note SCCP is a Cisco propreitary protocol. Special approval and technology licensing agreements are required in order to receive access to SCCP development resources.

Troubleshooting Tools

Commercially-available tools, such as Sniffer Pro or Ethereal, need to be used for collecting the packets exchanged between the skinny client and the Cisco CallManager server. The skinny client uses TCP/IP for transport. While configuring the network analysis software for capturing the packets, it is important to specify a filter between the the client and the Cisco CallManager sever.

Error Reporting

The Cisco CallManager traces can also be used to troubleshoot. Collect the Cisco CallManager CCM & SDL traces by setting them to detailed level. The section below details Cisco CallManager trace configuration.

Configuring Cisco CallManager Trace Parameters

This section describes how to configure trace parameters for the Cisco CallManager service.

Procedure

Step 1 From the Cisco CallManager Administration window, choose **Application** > **Cisco CallManager Serviceability**.

The Cisco CallManager Serviceability window displays.

Step 2 Choose **Trace** > **Configuration**.

Step 3 From the Servers column, choose the server.

The server that you chose displays next to the Current Server title and a box with configured services displays.

Step 4 From the Configured Services box, choose the Cisco CallManager service.

The service that you chose displays next to the Current Service title, along with the current server that you chose. The trace parameters display for the service that you chose.

Note Only the trace parameters for the service you chose are displayed. The display shows all other parameters grayed out.

Step 5 Check the *Trace On* checkbox.

Step 6 If you want trace to apply to all Cisco CallManager servers in the cluster, check the *Apply to All Nodes* checkbox.

Step 7 In the *Debug Trace Level* selection box, click the Down arrow.

A list with six debug trace levels displays.

Step 8 Click the desired debug trace level as described in Table 4.

Table 4 Debug Trace Levels

Level	Description
Error	Traces alarm conditions and events. Used for all traces generated in abnormal path. Uses minimum amount of CPU cycles.
Special	Traces all Error conditions plus process and device initialization messages.
State Transition	Traces all Special conditions plus subsystem state transitions that occur during normal operation. Traces call- processing events.
Significant	Traces all State Transition conditions plus media layer events that occur during normal operation.
Entry/Exit	Traces all Significant conditions plus entry and exit points of routines. Not all services use this trace level (for example, Cisco CallManager does not).
Arbitrary	Traces all Entry/Exit conditions plus low-level debugging information. **Note** Do not use this trace level with the Cisco CallManager service or the Cisco IP Voice Media Streaming Application service during normal operation.
Detailed	Traces all Arbitrary conditions plus detailed debugging information. **Note** Do not use this trace level with the Cisco CallManager service or the Cisco IP Voice Media Streaming Application service during normal operation.

Step 9 Check the *Cisco CallManager Trace Fields* check box. Table 5 describes the 17 options from which to choose.

Table 5 Cisco CallManager Trace Fields

Field Name	Description
Enable H245 Message Trace	Activates trace of H245 messages.
Enable DT-24+/DE-30+ Trace	Activates the logging of ISDN type of DT-24+/DE-30+ device traces.
Enable PRI Trace	Activates trace of primary rate interface (PRI) devices.
Enable ISDN Translation Trace	Activates ISDN message traces. Used for normal debugging.

Table 5 Cisco CallManager Trace Fields (continued)

Field Name	Description
Enable H225 & Gatekeeper Trace	Activates trace of H.225 devices. Used for normal debugging.
Enable Miscellaneous Trace	Activates trace of miscellaneous devices. **Note** Do not check this check box during normal system operation.
Enable Conference Bridge Trace	Activates trace of conference bridges. Used for normal debugging.
Enable Music on Hold Trace	Activates trace of music on hold (MOH) devices. Used to trace MOH device status such as registered with Cisco CallManager, unregistered with Cisco CallManager, and resource allocation processed successfully or failed.
Enable CM Real-Time Information Server Trace	Activates Cisco CallManager real-time information traces used by the real-time information server.
Enable CDR Trace	Activates traces for CDR.
Enable Analog Trunk Trace	Activates trace of all analog trunk (AT) gateways.
Enable All Phone Device Trace	Activates trace of phone devices. Trace information includes SoftPhone devices. Used for normal debugging.
Enable MTP Trace	Activates trace of media termination point (MTP) devices. Used for normal debugging.
Enable All Gateway Trace	Activates trace of all analog and digital gateways.
Enable Forward and Miscellaneous Trace	Activates trace for call forwarding and all subsystems not covered by another check box. Used for normal debugging.
Enable MGCP Trace	Activates trace for media gateway control protocol (MGCP) devices. Used for normal debugging.
Enable Media Resource Manager Trace	Activates trace for media resource manager (MRM) activities.

Step 10 If you want trace information for specific Cisco CallManager devices, check the De*vice Name Based Trace Monitoring* check box.

If you want trace to apply to non-devices in addition to devices, check the *Include Non-device Traces* check box. If check box is checked, set the appropriate debug trace level as described in Table 4.

Step 11 Check the *Enable File Trace Log* check box to enable the log file to receive trace information.

The default log filename and the default parameters display in the fields. If you want to send the trace information to another file, specify the filename and pathname by clicking the *filename* field. Change the default parameters by clicking the appropriate field and entering the information.

Note Trace validates the filename and ensures that the filename has a .txt extension. Do not use a filename that exists on another computer. Use a filename that exists on the computer that is running the trace.

The following default trace log filename applies for the Cisco CallManager: C:\ProgramFiles\Cisco\Trace\CCM\ccm.txt. See Table 6 for the trace log file default parameters.

Step 12 If you want the trace information to be available for trace analysis, check the *Enable XML Formatted Output* check box. If this check box is not checked, the log file compiles in text format and will not be available for trace analysis.

Step 13 If you are a Cisco engineer debugging the system, check the *Enable Debug Output String* check box; otherwise, continue with the following steps.

Step 14 To configure SDL Trace parameters, click **SDL Configuration**. See "Configuring SDL Trace Parameters" on page 49.

Step 15 To save your trace parameters configuration, click the **Update** button.

The changes to trace configuration take effect immediately for Cisco CallManager.

Note To set the default, click the **SetDefault** button. To apply the current settings for chosen services to all nodes in a cluster, check the *Apply to all Nodes* check box.

Table 6 Trace Log File Description

Field	Description
Maximum number of files	The total number of trace files for a given service. Cisco CallManager automatically appends a sequence number to the filename to indicate which file it is; for example, ccm299.txt. When the last file in the sequence is full, the trace data begins writing over the first file. The default is 300 files.
Maximum number of lines	The maximum number of lines of data stored in each trace file. The default is 10000 lines for text files and 2000 for XML files.
Maximum number of minutes	The maximum minutes of data stored in each trace file. The default is 1440 minutes.

When the trace data exceeds either the maximum number of lines or the maximum minutes for one file, Cisco CallManager closes that file and writes the remaining trace data to the next file in the sequence. For example, you can set up trace files to store a full week of data, with one day of data in each file. To do this, set the number of files to 7, the minutes to 1440 (one day), and the number of lines to a large value such as 10000 (or larger for a busy system).

Configuring SDL Trace Parameters

This section describes how to configure the SDL trace parameters for the Cisco CallManager and Cisco CTIManager services.

Procedure

Step 1 From the Cisco CallManager or Cisco CTIManager trace configuration window, click the **SDL Configuration** link.

The SDL Trace Configuration window displays.

Step 2 Check the *Trace On* check box.

Step 3 If you are configuring SDL parameters for the Cisco CallManager service, check the *Trace Filter Settings* check boxes that you want to apply to this trace as described in Table 7. If you are configuring the SDL parameters for the Cisco CTIManager service, check the *Trace Filter Settings* check boxes that you want to apply to this trace as described in Table 8.

Note Cisco recommends that you use the defaults unless a Cisco engineer instructs you to do otherwise.

Table 7 **Cisco CallManager SDL Configuration Filter Settings**

Setting Name	Description
Enable All Layer 1 Trace	Activates traces for Layer 1.
Enable Detailed Layer 1 Trace	Activates detailed Layer 1 traces.
Enable All Layer 2 Trace	Activates traces for Layer 2.
Enable Layer 2 interface Trace	Activates Layer 2 interface traces.
Enable Layer 2 TCP Trace	Activates Layer 2 Transmission Control Program (TCP) traces.
Enable Detailed Dump Layer 2 Trace	Activates detailed traces for dump Layer 2.
Enable All Layer 3 Trace	Activates traces for Layer 3.
Enable All Call Control Trace	Activates traces for call control.
Enable Miscellaneous Polls Trace	Activates traces for miscellaneous polls.
Enable Miscellaneous Trace (Database Signals)	Activates miscellaneous traces such as database signals.
Enable Message Translation Signals Trace	Activates traces for message translation signals.
Enable UUIE Output Trace	Activates traces for user-to-user informational element (UUIE) output.
Enable Gateway Signals Trace	Activates traces for gateway signals.
Enable CTI Trace	Activates CTI trace.
Enable Network Service Data Trace	Activates network service data trace.
Enable Network Service Event Trace	Activates network service event trace.
Enable ICCP Admin Trace	Activates admin trace for ICCP.
Enable Default Trace	Activates default trace.

Table 8 Cisco CTIManager Trace SDL Configuration Filter Settings

Setting Name	Description
Enable Miscellaneous Polls Trace	Activates traces for miscellaneous polls.
Enable Miscellaneous Trace (Database Signals	Activates miscellaneous traces such as database signals.
Enable CTI SDL Trace	Activates CTI SDL trace.
Enable CTI Application Trace	Activates trace for CTI applications.
Enable CTI Information Trace	Activates trace for CTI information.
Enable CTI Warning Trace	Activates warning trace for CTI.
Enable CTI Error Trace	Activates error trace for CTI.
Enable Network Service Data Trace	Activates network service data trace.
Enable Network Service Event Trace	Activates network service event trace.
Enable ICCP Admin Trace	Activates admin trace for ICCP.
Enable Default Trace	Activates default trace.

Step 4 If you are configuring SDL parameters for the Cisco CallManager service, check the *Trace Characteristics* check boxes that you want to apply to this trace as described in Table 9. If you are configuring the SDL parameters for the Cisco CTIManager service, check the *Trace Characteristics* check boxes that you want to apply to this trace as described in Table 10.

Table 9 Cisco CallManager SDL Configuration Trace Characteristics

Characteristics	Description
Enable SDL Link States Trace	Activates trace for intracluster communication protocol (ICCP) link state.
Enable Low-Level SDL Trace	Activates trace for low-level SDL.
Enable SDL Link Poll Trace	Activates trace for ICCP link poll.
Enable SDL Link Messages Trace	Activates trace for ICCP raw messages.
Enable Signal Data Dump Trace	Activates traces for signal data dump.
Enable Correlation Tag Mapping Trace	Activates traces for correlation tag mapping.
Enable SDL Process States Trace	Activates traces for SDL process states.
Disable Pretty Print of SDLTrace	Disables trace for pretty print of SDL. Pretty print adds tabs and spaces in a trace file without performing post processing.
Enable SDL TCP Event Trace	Activates trace for SDL TCP event.

Table 10 Cisco CTIManager SDL Configuration Characteristics

Characteristics	Description
Enable SDL Link States Trace	Activates trace for ICCP link state.
Enable Low-level SDL Trace	Activates trace for low-level SDL.
Enable SDL Link Poll Trace	Activates trace for ICCP link poll.
Enable SDL Link Messages Trace	Activates trace for ICCP raw messages.
Enable Signal Data Dump Trace	Activates traces for signal data dump.
Enable Correlation Tag Mapping Trace	Activates traces for correlation tag mapping.
Enable SDL Process States Trace	Activates traces for SDL process states.
Disable Pretty Print of SDL Trace	Disables trace for pretty print of SDL. Pretty print adds tabs and spaces in a trace file without performing post processing.
Enable SDL TCP Event Trace	Activates trace for SDL TCP event.

Step 5 If you want the trace information available for trace analysis, check the *Enable XML Formatted Output* check box. If this check box is not checked, the log file compiles in text format and will not be available for trace analysis.

The default trace directory path and the default parameters display in the fields. If you want to send the trace information to another file, enter the filename and pathname in the *Trace Directory Path* field. Change the default parameters by clicking the appropriate field and entering the information.

The following default trace log filename applies for SDL Trace Configuration: C:\Program Files\Cisco\Trace\SDL. See Table 6 for the Trace log file default parameters.

Step 6 To save your SDL trace parameters configuration, click the **Update** button.

The changes to trace configuration take effect immediately for SDL Trace Configuration.

Note To set the default, click the SetDefault button. To apply the current settings for chosen services to all nodes in a cluster, check the Apply to all Nodes check box.

Step 7 To continue with SDL Trace Configuration for another service, choose the service from the *Configured Services* box; otherwise, continue with Step 8.

Step 8 To return to the Cisco CallManager or Cisco CTIManager SDI Trace Configuration window, click the **SDI Configuration** link.

Cisco CallManager Express and Survivable Remote Site Telephony Troubleshooting

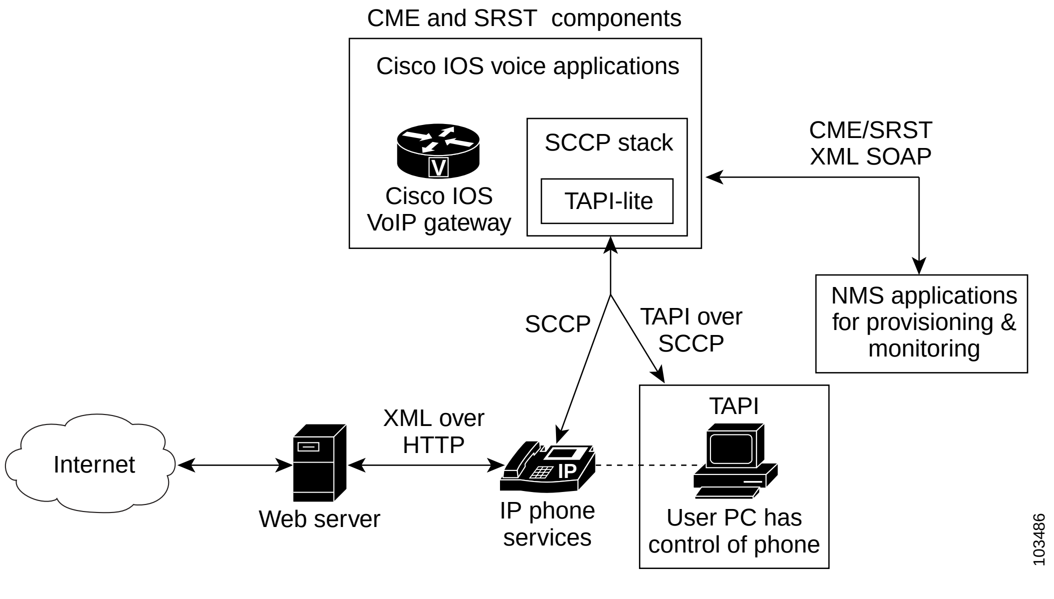
This chapter contains the following topics:

- Overview, page 53
- PostInstallation Checklist for TAPILite, page 54
- Debug Traces, page 54

Overview

Figure 28 shows an architecture overview of Cisco CallManager Express (CME) and Cisco Survivable Remote Site Telephony (SRST).

Figure 28 Cisco CallManager Express and SRST Architecture Overview



Cisco CME allows small business customers and enterprise branch offices to deploy voice, data, and IP telephony on a single platform. Cisco CME supports the following APIs in addition to supporting Tcl and VoiceXML:

- SCCP— Cisco CME uses the Skinny Client Control Protocol (SCCP) to control Cisco IP phones.
- RouterXML— An extension of the Cisco AVVID XML Layer (AXL) API layer that provides a mechanism for configuring, provisioning, and monitoring Cisco CME routers.
- TAPILite (A version of TAPI)— The TAPILite layer is a lighter implementation of the full TAPI stack. It consists of a set of classes that help developers create customized telephony applications for Cisco CME.

Cisco SRST provides Cisco CallManager with fallback support for Cisco IP phones if a WAN failure is detected. Cisco SRST runs under the Cisco IOS stack and uses the Skinny protocol (SCCP) to interact with Cisco IP phones.

PostInstallation Checklist for TAPILite

Some problems may occur because of improper installation. Use this checklist to verify proper installation.

____ Ensure that Cisco CallManager is running with the correct configuration for TAPI client.

____ Ensure that the network connection is up between the Telephony Service Provider (TSP) and Cisco CallManager Express (CME).

____ Ensure that the username and password in the TSP match the username and password in Cisco CME.

____ Ensure that all devices are correctly associated with the user in Cisco CME.

____ Ensure that the Cisco TSP is the correct version.

Debug Traces

Use the **debug ephone detail** command to debug the output. “Capturing a Debug Log” on page 54 outlines the steps required to capture a debug log using the logging buffer method. The debug traces that are analyzed in “Analyzing Traces” on page 55 are captured from generic scenarios and are provided as a general road map for troubleshooting.

If the problem cannot be resolved, contact the Cisco TAC or Developer Support. For more information on opening a case with the TAC or Developer Support, see “General Troubleshooting Procedures” on page 1.”

Capturing a Debug Log

If you use a logging console to capture debug output, you may not be able to capture the complete log of the output. To ensure that you capture the complete log of your debug output, use the logging buffer method as shown below:

Step 1 Configure the logging buffer.

```
Router# configure terminal
Router(config)# no logging console
Router(config)# logging buffer 2000000
Router(config)# end
```

Step 2 Clear the logging buffer.

```
Router# clear logging
```

Step 3 Verify the Cisco IOS release on the router, and run the configuration.

```
Router# show version
Router# show run
```

Step 4 Turn on the debugs and reproduce the problem.

Step 5 Capture the debug output to a file after entering the **show log** command.

```
Router# show log
```

Analyzing Traces

This section consists of sample debug output for:

- The Skinny Client Control Protocol (SCCP) for Cisco IP Phone registration with Cisco SRST and Cisco CME.
- TAPILite client registration with Cisco CME.

Skinny Client Control Protocol

SRST

Following is a sample debug trace of **show ephone detail** where the Cisco IP phone registers with Cisco SRST after detecting a failure in Cisco CallManager.

```
Router# debug ephone detail
EPHONE detail debugging is enabled
*May 31 11:05:43.584: %IPPHONE-6-REG_ALARM: 13: Name=SEP000D287E44A9 Load=6.0(0.1)
Last=CM-aborted-TCP
*May 31 11:05:43.584: ephone-(3)[2] StationRegisterMessage (0/0/8) from 13.13.13.16
*May 31 11:05:43.584: ephone-(3)[2] Register StationIdentifier DeviceName SEP000D287E44A9
*May 31 11:05:43.584: ephone-3[2]:stationIpAddr 13.13.13.16
*May 31 11:05:43.584: ephone-3[2]:phone SEP000D287E44A9 re-associate OK on socket [2]
*May 31 11:05:43.584: %IPPHONE-6-REGISTER_NEW: ephone-3:SEP000D287E44A9 IP:13.13.13.16
Socket:2
DeviceType:Phone has registered.
*May 31 11:05:43.841: ephone-3[2]:InterogatePhone
*May 31 11:05:43.841: ephone-3[2]:Sending StationServerReqMessage
<= Cisco SRST contacts the IP phone and requests server information
*May 31 11:05:44.098: Server List:
*May 31 11:05:44.098: CM1 13.13.13.50
*May 31 11:05:44.098: CM2 0.0.0.0
<snip>
*May 31 11:05:44.351: ephone-3[2][SEP000D287E44A9]:ButtonTemplate offset 0 buttons 6 total
6
<= Cisco SRST receives button template from the IP Phone
*May 31 11:0
DevSup-BO-26515:44.351: ephone-3[2]:Button Type 9
*May 31 11:05:44.351: ephone-3[2]:Found Line Button 1
*May 31 11:05:44.351: ephone-3[2]:Button Type 9
*May 31 11:05:44.351: ephone-3[2]:Found Line Button 2
*May 31 11:05:44.351: ephone-3[2]:Button Type 2
```

```
<snip>
*May 31 11:05:44.351: ephone-3[2]:Found Speed Dial Button 6
*May 31 11:05:44.351: ephone-3[2]:Requesting first LineStatReq
*May 31 11:05:44.608: ephone-3[2][SEP000D287E44A9]:SkinnyTryCall to 2000 instance 1 start
at 0

*May 31 11:05:44.608: Skinny Update dn 1 chan 2
*May 31 11:05:44.608: ephone-3[2]:SkinnyAllocateDN 1 2000 2000
*May 31 11:05:44.608: ephone-3[2]:
LineStatMessage Auto Create new DN 1 line 1
*May 31 11:05:44.608: ephone-3[2]:
LineStatMessage Auto Bind new DN 1 to line 1
*May 31 11:05:44.608: Skinny DN 1 chan 1 state change to UP
*May 31 11:05:44.612: Skinny DN 1 chan 2 state change to UP
*May 31 11:05:44.865: ephone-3[2]:Auto Bind with null phone number for line 2
*May 31 11:05:45.118: ephone-3[2]:Auto set with null speedDial 1
*May 31 11:05:45.371: ephone-3[2]:Auto set with null speedDial 2
<snip>
*May 31 11:05:45.877: ephone-3[2]:
SpeedDialStatMessage Auto Set completed 2 lines 4 speed
*May 31 11:05:46.130: ephone-3[2]:SoftKeys ver: 3| 1 tag 1
*May 31 11:05:46.130: ephone-3[2]:SoftKeys ver: 3| 2 tag 2
<snip>
*May 31 11:05:47.905: ephone-3[2]:ClearCallPrompt line 0 ref 0
*May 31 11:05:47.905: ephone-3[2]:SelectPhoneSoftKeys set 0 mask 7 for line 0 ref 0
*May 31 11:05:47.905: ephone-3[2][SEP000D287E44A9]:CallPrompt line 0 ref 0: CM Fallback
Service Operating
<= Cisco SRST requests the IP Phone to display the "CM Fallback Service Operating" message

*May 31 11:05:47.909: ephone-3[2]:SkinnyCheckPendingCallBackPhone scan 6 lines
*May 31 11:05:47.909: ephone-3[2][SEP000D287E44A9]: DN in-service for DN 1 chan 1
*May 31 11:05:47.909: SkinnyGetCallState for DN 1 chan 1 IDLE
*May 31 11:05:47.909: called DN -1 chan 1, calling DN -1 chan 1 phone -1 s2s:0
*May 31 11:05:47.909: ephone-3[2][SEP000D287E44A9]: DN in-service for DN 1 chan 2
*May 31 11:05:47.909: SkinnyGetCallState for DN 1 chan 2 IDLE
*May 31 11:05:47.909: called DN -1 chan 1, calling DN -1 chan 1 phone -1 s2s:0
*May 31 11:05:47.921: ephone-3[2][SEP000D287E44A9]:CallPrompt line 0 ref 0: SRST
*May 31 11:05:47.921: ephone-3[2][SEP000D287E44A9]:CallPrompt line 0 ref 0: CM Fallback
Service Operating
*May 31 11:05:47.921: ephone-3[2]:SkinnyCheckPendingCallBackPhone scan 6 lines
*May 31 11:05:47.925: ephone-3[2]:TimeDateReqMessage
<= Cisco SRST receives time date request from the IP phone.

*May 31 11:05:47.925: year=1993 month=5 day=31
*May 31 11:05:47.925: hour=11 minute=5 second=47
*May 31 11:05:47.925: day=1 dayofyear=151 tzoffset=0
*May 31 11:05:47.925: ephone-3[2]:DefineTimeDate sent#
<=Cisco SRST sends the time date to the IP phone

Router#
*May 31 11:05:57.538: ephone-3[2]:Check Hot-Sync for 6 out of 1 lines
*May 31 11:05:57.538: SkinnyGetCallState for DN 1 chan 1 IDLE
*May 31 11:05:57.538: called DN -1 chan 1, calling DN -1 chan 1 phone -1 s2s:0
*May 31 11:05:57.538: ephone-3[2]:SetCallState line 1 DN 1 chan 1 ref 0 TsOnHook
*May 31 11:05:57.538: ephone-3[2]:ClearCallPrompt line 1 ref 0
*May 31 11:05:57.538: ephone-3[2]:SelectPhoneSoftKeys set 0 mask 7 for line 1 ref 0
*May 31 11:05:57.538: SkinnyGetCallState for DN 1 chan 2 IDLE
*May 31 11:05:57.538: called DN -1 chan 1, calling DN -1 chan 1 phone -1 s2s:0
*May 31 11:05:57.542: ephone-3[2]:SetCallState line 1 DN 1 chan 2 ref 0 TsOnHook
Router#
*May 31 11:05:57.542: ephone-3[2]:ClearCallPrompt line 1 ref 0
*May 31 11:05:57.542: ephone-3[2]:SelectPhoneSoftKeys set 0 mask 7 for line 1 ref 0
*May 31 11:05:57.542: ephone-3[2]:Set MWI line 1 to OFF
*May 31 11:05:57.542: ephone-3[2]:Set MWI line 0 to OFF
```

```
*May 31 11:05:57.546: ephone-3[2][SEP000D287E44A9]:CallPrompt line 0 ref 0: CM Fallback
Service Operating
*May 31 11:05:57.546: ephone-3[2]:SkinnyCheckPendingCallBackPhone scan 6 lines
Router#
```

Following is a sample debug trace of **show ephone detail** for Skinny phone registration with Cisco SRST where the Cisco IP Phone un-registers with Cisco SRST after it detects that Cisco CallManager is back in service.

```
Router# debug ephone detail
EPHONE detail debugging is enabled

*May 31 11:06:24.110: ephone-3[2][SEP000D287E44A9]: DN out-of-service for DN 1 chan 1
*May 31 11:06:24.110: SkinnyGetCallState for DN 1 chan 1 IDLE
<= Cisco SRST requests the call state for IP phone with DN 1 and channel 1.

*May 31 11:06:24.110: called DN -1 chan 1, calling DN -1 chan 1 phone -1 s2s:0
*May 31 11:06:24.110: ephone-3[2][SEP000D287E44A9]: DN out-of-service for DN 1 chan 2
<=Cisco SRST requests the call state for IP phone with DN 1 and channel 2.

*May 31 11:06:24.110: SkinnyGetCallState for DN 1 chan 2 IDLE
*May 31 11:06:24.110: called DN -1 chan 1, calling DN -1 chan 1 phone -1 s2s:0
*May 31 11:06:24.110: Skinny DN 1 chan 1 state change to DOWN
*May 31 11:06:24.110: Skinny DN 1 chan 2 state change to DOWN
*May 31 11:06:24.110: Skinny set DN number  on dn 1
*May 31 11:06:24.114: Skinny set DN name  on dn 1
*May 31 11:06:24.114: ephone-3[2]:SkinnyFreeDN 1
*May 31 11:06:24.114: ephone-3[2]:UnregisterAck sent on socket [2] (0/0/8)
<= Cisco SRST sends an unregistered acknowledgement message to the IP phone.

Router #
*May 31 11:06:24.114: %IPPHONE-6-UNREGISTER_NORMAL: ephone-3:SEP000D287E44A9
IP:13.13.13.16 Socket:2


DeviceType:Phone has unregistered normally.
<= Cisco SRST unregisters from Cisco CallManager.
Router#
```

Cisco CME

Following is a sample debug trace of **debug ephone detail** for Skinny phone registration with Cisco CME where a Cisco IP phone with dn 3000 registers with Cisco CME.

```
Router#debug ephone detail
EPHONE detail debugging is enabled
Router#
*Apr 20 00:29:34.251: ephone-(4)[1] StationRegisterMessage (3/7/20) from 200.0.0.3
<= Cisco CME receives the registration request.

*Apr 20 00:29:34.251: ephone-(4)[1] Register StationIdentifier DeviceName SEP000D287E44A9
*Apr 20 00:29:34.251: ephone-4[-1]:stationIpAddr 200.0.0.3
*Apr 20 00:29:34.251: ephone-4[-1]:socket change -1 to 1
*Apr 20 00:29:34.251: ephone-4[1]:phone SEP000D287E44A9 re-associate OK on socket [1]

*Apr 20 00:29:34.764: ephone-4[1]:SoftKeys ver: 3| 11
*Apr 20 00:29:34.764: ephone-4[1]:SoftKeys ver: 3| 12
<snip>
*Apr 20 00:29:35.537: ephone-4[1]:CallPrompt line 0 ref 0: ITS
*Apr 20 00:29:35.790: ephone-4[1]:Speed Dial Label 3 : PAGING ALL
```

```
*Apr 20 00:29:36.042: ephone-4[1]:Speed Dial Label 2 : PAGING BR
*Apr 20 00:29:36.294: ephone-4[1]:Speed Dial Label 1 : PAGING HQ
*Apr 20 00:29:36.547: ephone-4[1]:TimeDateReqMessage
<= Cisco CME receives the time date request

*Apr 20 00:29:36.547: year=1993 month=4 day=20
*Apr 20 00:29:36.547: hour=0 minute=29 second=36
*Apr 20 00:29:36.547: day=2 dayofyear=110 tzoffset=0
*Apr 20 00:29:36.547: ephone-4[1]:DefineTimeDate sent
<= Cisco CME sends the time date request to the IP phone.

*Apr 20 00:29:44.624: ephone-4[1]:ClearCallPrompt line 0 ref 0
*Apr 20 00:29:44.624: ephone-4[1]:SelectPhoneSoftKeys set 0 for line 0 ref 0
*Apr 20 00:29:44.624: ephone-4[1]:CallPrompt line 0 ref 0:  Cisco ITS
<= Cisco CME notifies the IP phone to display "Cisco ITS". Cisco ITS is the old name for
Cisco SRST.

*Apr 20 00:29:44.624: ephone-4[1]: DN in-service for DN 4
*Apr 20 00:29:44.624: SkinnyGetCallState for DN 4 IDLE
*Apr 20 00:29:44.624: called DN -1, calling DN -1 phone -1 s2s:0
*Apr 20 00:29:44.624: Skinny DN 4 state change to UP
*Apr 20 00:29:44.624: ephone-4[1]: DN in-service for DN 11
*Apr 20 00:29:44.624: SkinnyGetCallState for DN 11 IDLE
*Apr 20 00:29:44.628: called DN -1, calling DN -1 phone -1 s2s:0
*Apr 20 00:29:44.628: Skinny DN 11 state change to UP
*Apr 20 00:29:44.628: ephone-4[1]:Check Hot-Sync for 6 out of 2 lines
*Apr 20 00:29:44.628: SkinnyGetCallState for DN 4 IDLE
*Apr 20 00:29:44.628: called DN -1, calling DN -1 phone -1 s2s:0
*Apr 20 00:29:44.628: ephone-4[1]:SetCallState line 1 DN 4 ref 32 TsOnHook
<= Cisco CME notifies the IP phone that it is in the ON HOOK state for line 1 DN4.

*Apr 20 00:29:44.628: ephone-4[1]:ClearCallPrompt line 1 ref 32
<= Cisco CME notifies the IP phone to clear the prompt.

*Apr 20 00:29:44.628: ephone-4[1]:SelectPhoneSoftKeys set 0 for line 1 ref 32
<= Cisco CME notifies the IP phone to update softkey mapping for line 1.

*Apr 20 00:29:44.632: SkinnyGetCallState for DN 11 IDLE
*Apr 20 00:29:44.632: called DN -1, calling DN -1 phone -1 s2s:0
*Apr 20 00:29:44.632: ephone-4[1]:SetCallState line 2 DN 11 ref 13 TsOnHook
<= Set line 2


DN 11 to status onhook
*Apr 20 00:29:44.632: ephone-4[1]:ClearCallPrompt line 2 ref 13
*Apr 20 00:29:44.632: ephone-4[1]:SelectPhoneSoftKeys set 0 for line 2 ref 13
<= Cisco CME notifies the IP phone to update softkey mapping for line 2.

*Apr 20 00:29:44.632: ephone-4[1]:Set MWI line 1 to OFF
*Apr 20 00:29:44.636: ephone-4[1]:Set MWI line 0 to OFF
*Apr 20 00:29:44.636: ephone-4[1]:Set MWI line 2 to OFF
Router
```

TAPILite

Following is a sample debug trace of **debug ephone detail** for TAPILite client registration with Cisco CME where a Cisco IP phone with dn 3000 registers with Cisco CME.

```
Router# Register Message From Tapi Client
```

```
*Apr 20 01:44:48.511: ephone-1[3]:Register Message From TAPI Client 0
<= Cisco CME received registration from TAPI client.

*Apr 20 01:44:48.511: ephone-1[3]:socket change -1 to 5
*Apr 20 01:44:48.511: ephone-1[3]:socket 5 is associated to a TAPI Client 0
*Apr 20 01:44:48.511: ephone-1[3]:Username and password verified
<= TAPILite Client successfully verified for TAPI Client.

*Apr 20 01:45:08.704: ephone-1[3]:Check Hot-Sync for 6 out of 2 lines
*Apr 20 01:45:08.704: SkinnyGetCallState for DN 6 IDLE
*Apr 20 01:45:08.704: called DN -1, calling DN -1 phone -1 s2s:0
*Apr 20 01:45:08.704: ephone-1[3]:SetCallState line 2 DN 6 ref 85 TsOnHook
<= Sets line 2 DN6 to status onhook.

*Apr 20 01:45:08.704: ephone-1[3]:ClearCallPrompt line 2 ref 85
*Apr 20 01:45:08.704: ephone-1[3]:ClearCallPrompt line to Tapi Client on line 2 ref 85
<= Sends clear prompt request to TAPILite Client.

*Apr 20 01:45:08.708: ephone-1[3]:SelectPhoneSoftKeys set 0 for line 2 ref 85
*Apr 20 01:45:08.708: ephone-1[3]:Set MWI line 2 to OFF
Router#
```

GLOSSARY

This chapter lists common terms and acronyms used throughout this document. For a more detailed list of internetworking terms and acronyms, refer to the Internetworking and Acronyms web site at:

http://www.cisco.com/univercd/cc/td/doc/cisintwk/ita/index.htm

A

ASP	Active server page. A web page that uses ActiveX scripting to dynamically control the content of the web page. Cisco CallManager Administration relies on Active server pages.

J

JSP	Java server page. A web page that uses Java scripting to dynamically control the content of the web page.
JTAPI	Java Telephony Applications Programming Interface

P

parser	A program that breaks an input stream into syntactic elements. Cisco IP Phones have an XML parser that breaks out individual element values for the phone’s firmware.
PULL	In XML Services troubleshooting, the sending of an XML object to an IP phone as a result of a request made at the phone.
PUSH	In XML Services troubleshooting, the sending of an XML object to an IP phone as a result of the web service initiating content to be sent to the phone.

R

RTP	Real-Time Transport Protocol. A User Datagram Protocol (UDP)-based protocol especially designed to send voice, video, and other time-sensitive data across packet networks.

S

SCCP	Skinny Client Control Protocol. Cisco-proprietary protocol that is used to send messages between Cisco CallManager and an IP phone.

SDK	Software Development Kit. A set of programming interfaces and documentation provided to programmers seeking to interface to a given operating system, application, or other product.
SGML	Standard Generalized Markup Language. A universal language for creating device-independent documents with useful formatting and content.

T

TAPI	Telephony Applications Programming Interface
TSP	Telephony Service Provider

V

Validator	A program that checks Cisco IP Phone XML objects for validity by parsing the objects and comparing the syntactic elements against the XML schema. This program is available on the CD-ROM provided with the Cisco Press book, *Developing Cisco IP Phone Services*.

X

XML	Extensible Markup Language. A simple dialect of SGML that enables generic SGML to be served, received, and processed on the web. Also, the universal format for structured documents and data on the web.
XML schema	Document that defines which tags can be used where, what type of content can compose an element, and the order in which elements can appear. XML schemas also define acceptable content.
XML Services	Applications for IP phones in Cisco CallManager environments.