



**Installation, Configuration,
Troubleshooting, and
Programmer's Reference for
Avaya Aura™ Communication
Manager Branch application
enablement (CTI)**

03-602030
Issue 3
May 2009

Notice

While reasonable efforts were made to ensure that the information in this document was complete and accurate at the time of printing, Avaya Inc. can assume no liability for any errors. Changes and corrections to the information in this document may be incorporated in future releases.

For full support information, please see the complete document, *Avaya Support Notices for Software Documentation*, document number 03-600758.

To locate this document on our Web site, simply go to <http://www.avaya.com/support> and search for the document number in the search box.

Documentation disclaimer

Avaya Inc. is not responsible for any modifications, additions, or deletions to the original published version of this documentation unless such modifications, additions, or deletions were performed by Avaya. Customer and/or End User agree to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation to the extent made by the Customer or End User.

Link disclaimer

Avaya Inc. is not responsible for the contents or reliability of any linked Web sites referenced elsewhere within this documentation, and Avaya does not necessarily endorse the products, services, or information described or offered within them. We cannot guarantee that these links will work all of the time and we have no control over the availability of the linked pages.

Warranty

Avaya Inc. provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya's standard warranty language, as well as information regarding support for this product, while under warranty, is available through the following Web site: <http://www.avaya.com/support>.

Copyright

Except where expressly stated otherwise, the Product is protected by copyright and other laws respecting proprietary rights. Unauthorized reproduction, transfer, and or use can be a criminal, as well as a civil, offense under the applicable law.

Avaya support

Avaya provides a telephone number for you to use to report problems or to ask questions about your product. The support telephone number is 1-800-242-2121 in the United States. For additional support telephone numbers, see the Avaya Web site: <http://www.avaya.com/support>.

Contents

Chapter 1: About Avaya Aura Communication Manager Branch application enablement	7
What is Avaya Aura Communication Manager Branch application enablement? .	7
Which applications run with Communication Manager Branch application enablement?	8
Who should use this document?	8
 Chapter 2: Overview of Avaya Aura Communication Manager Branch application enablement	 9
Backward Compatibility	9
Architecture	9
Communication Manager Branch application enablement architecture diagram	9
Avaya Aura Communication Manager Branch application enablement software	10
Core software	11
Avaya CTI client software	11
Avaya CTI client software installer	12
Application computers	12
Client hardware requirements	13
Limitations	14
Capacities	15
UI Support	15
 Chapter 3: Communication Manager Branch Server CTI Configuration.	 17
CTI User Account authentication	17
Local CTI user account authentication.	18
Microsoft Active Directory Services for user account authentication	19
Creating a CTI User Account on the Domain Controller	20
Generating a keytab file for the account on the Domain Controller	20
Installing the keytab file and enabling Active Directory authentication in Branch Device Manager	22
Creating CTI Extensions	23
 Chapter 4: Installing and Configuring the Avaya CTI Software on each Application Computer.	 27
Introduction	27
Installing the Avaya TAPI TSP	28
Preinstallation tasks.	28
Installation procedure	29
Configuring the Avaya TAPI TSP	30

Running the Avaya TAPI TSP installation test	35
Removing the Avaya TAPI TSP	36
Reinstalling the Avaya TAPI TSP	36
Installing and Configuring TSAPI Client Software.	37
Preinstallation Tasks	38
Installing and Configuring TSAPI Client Software on Linux Operating Systems	39
Installing and Configuring TSAPI Client Software on Windows XP	40
Installing and Configuring TSAPI Client Software on Other Systems	42
Chapter 5: Managing Avaya Aura Communication Manager Branch application enablement	45
Logging CTI errors and exceptions	45
Adding and deleting local CTI user accounts for authentication.	45
Monitoring CTI status	46
Chapter 6: Troubleshooting CTI Problems	49
Troubleshooting Problems with TAPI-based CTI	49
CTI TAPI-based applications are not functioning	49
CTI TAPI-based applications respond slowly to TAPI requests	58
CTI TAPI-based applications exhibit a specific telephony feature failure	59
Troubleshooting Problems with TSAPI-based CTI	59
TSAPI-based CTI applications are not functioning	60
TSAPI-based CTI applications respond slowly to TSAPI requests.	65
TSAPI-based CTI applications exhibit a specific telephony feature failure	65
Chapter 7: Programmer's Reference Information	67
General Characteristics of Communication Manager Branch CTI Support	67
Multi-application interaction	67
CTI-controllable endpoints	68
SIP devices	68
Key configuration endpoints	68
CTI Extensions.	69
CTI-affecting Communication Manager Branch characteristics	69
CTI-specific alarms on the Communication Manager Branch server	70
Reliability and recovery	70
Using TAPI	70
Supported TAPI capability classes	71
Supported TAPI service requests	71

Supported TAPI events	73
Supported TAPI data structures	77
Using TSAPI	77
Introduction	77
Non-supported TSAPI Services and Events	78
Initiating and managing a TSAPI CTI session - ACS services and events. . .	80
Control and environment discovery services and events	82
Call control services and events	82
Logical feature services.	84
Query services	84
Snapshot services	84
Monitor services and events	85
System status services and events	85
Universal failure events	85
Appendix A: Tested Applications	87
Appendix B: Supported TAPI Services and Events	89
Supported data structure parameters	93
Appendix C: CTI Programming Considerations, Design Tips, and Best Practices	103
Collective Ringing Groups	104
Incoming call to CRG at a monitored member station that answers.	104
Incoming call to CRG at a monitored member station that is answered elsewhere	104
Incoming call to CRG at a monitored member station that is dropped before answer.	105
Outside Line Group Feature Behavior	105
Incoming call on OLG line.	105
Incoming call to OLG is answered by one station and other station connections transition to null	106
Incoming answered call to monitored OLG member on Line Appearance and additional OLG members subsequently join call	106
Incoming, answered call to monitored OLG member on Line Appearance and additional OLG members subsequently join call	108
Second OLG station to join OLG call hangs up. The first monitored station that originally answered (still on the call) gets TWO Connection Cleared events	109
Snapshot call does not always show all devices: example, when an OLG member hangs up and the call continues with other OLG member(s)	111
Last OLG Station Clears From Call - Multiple Connection Cleared Events	

Contents

Received 112

A CTI-Extension that is a member of an OLG cannot answer/join an OLG
call that is already active at one or more other "real" members
of the same OLG 114

No Connection Cleared Event from Outgoing OLG Call 114

Multiple Conferenced and Transferred events. 115

Appendix D: About the Avaya DevConnect Program 119

Glossary 121

Index 123

Chapter 1: About Avaya Aura Communication Manager Branch application enablement

What is Avaya Aura Communication Manager Branch application enablement?

Avaya Aura™ Communication Manager Branch application enablement provides computer telephony integration (CTI) for basic telephony at an Communication Manager Branch location. Communication Manager Branch supports:

- A subset of Microsoft Telephony Application Programming Interface (TAPI) services and events.
- A complete Avaya implementation of the Telephony Services Application Programming Interface (TSAPI) specification.

Communication Manager Branch enables CTI capabilities such as click-to-dial, call logging, and basic call control (transfer, conference, hold/unhold) used in common CTI-enabled desktop and server-based applications. This CTI support does not depend on any other facilities outside the Communication Manager Branch location.

Communication Manager Branch application enablement supports configurations in which an application on a application computer can control and/or monitor any or all of the telephone extensions in a branch. Application computers are the computers on which TAPI-based and TSAPI-based CTI-enabled applications are installed and on which you must install the Avaya TAPI TSP and/or TSAPI client software.

All Communication Manager Branch telephones are supported for CTI except as noted in [Limitations](#) on page 14.

The core software for Communication Manager Branch application enablement is already installed on the Communication Manager Branch platform. You must install and configure additional CTI client software (the Avaya Application Enablement Services TAPI Telephony Service Provider - Avaya TSP - and/or TSAPI) on each CTI-enabled application computer.

Which applications run with Communication Manager Branch application enablement?

[Appendix A: Tested Applications](#) on page 87 lists some popular TAPI-based applications that have been tested and found to work with Communication Manager Branch. If you intend to use other TAPI-based applications, check [Appendix B: Supported TAPI Services and Events](#) on page 89 to see if that application uses only the supported TAPI services and events.

If you intend to use or develop TSAPI-based applications, review the TSAPI sections of this document to verify that all required services and events are supported by Communication Manager Branch.

Note that even if all required TAPI or TSAPI services and events are supported, an existing application may behave differently with Communication Manager Branch than it does with other communications equipment. For example, an application might expect certain specific behavior from the equipment (such as the number of parties that can be added to a conference or specific event flow sequences) that differs from Communication Manager Branch capabilities. The TAPI and TSAPI Chapters of this document provide additional details on Communication Manager Branch's CTI behavior that will be useful in determining how to write new CTI-enabled application software and how existing CTI-enabled applications will operate with Communication Manager Branch.

Who should use this document?

Use this book if you are:

- An ISV, programmer or other IT professional who wants to better understand Communication Manager Branch's CTI capabilities and initially evaluate whether an existing CTI-enabled application can be ported to Communication Manager Branch.
- A technician who installs and configures the Communication Manager Branch application enablement client software and/or configures CTI features on the Communication Manager Branch server platform.
- A technician who installs and configures the Avaya Application Enablement Services TAPI Telephony Service Provider software and/or configures the CTI user authentication for Communication Manager Branch application enablement.
- A technician who troubleshoots CTI problems that arise during or after installation.
- A programmer who writes new TAPI or TSAPI based applications, integrates existing applications with Communication Manager Branch application enablement or ports existing TSAPI-based applications from Communication Manager to Communication Manager Branch environments.

Chapter 2: Overview of Avaya Aura Communication Manager Branch application enablement

Backward Compatibility

If you have upgraded from a previous release of Distributed Office, client software will continue to work with the Communication Manager Branch R2.0. No changes are required to continue using the TAPI software installed on each client application computer.

If you have not upgraded to Communication Manager Branch R2.0, Distributed Office Release 1.2 provides additional significant features - including some important security enhancements. Therefore you may wish to install the newer client software on each application computer. For example, in Release 1.2 the CTI messages between each client application computer and the Communication Manager Branch server can now be protected through the use of encrypted (secure) connections.

Note:

Before installing the new software on a client, you may need to remove existing software. Installation instructions in this document as well as the Client Software Installer wizard will guide you through any software removal process.

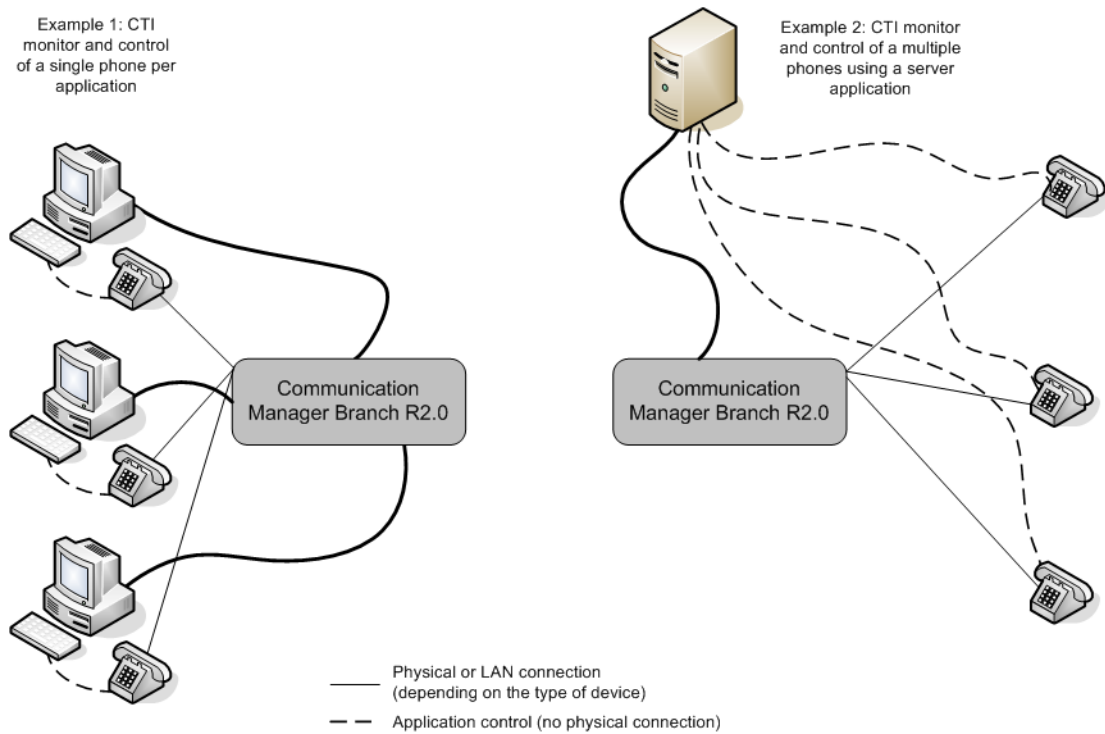
Architecture

Communication Manager Branch application enablement architecture diagram

Communication Manager Branch R2.0 can support two application configurations. In the personal desktop arrangement, the CTI-enabled application runs on a personal desktop computer and controls and monitors a single desktop phone.

[Figure 1, CTI Application running on a PC](#) illustrates the two different configuration. In Example 1, an employee has a click-to-dial application that automatically places calls to entries in his/her personal address book using their desktop phone. In Example 2, the application computer acts as a server, running on one application computer that controls and/or monitors multiple phones.

Figure 1: CTI Application running on a PC



Note:

The dotted lines illustrate which phones are being controlled by the CTI application using Communication Manager Branch's CTI features. There is no physical (LAN or hardwire) connection between the phone being controlled and the computer on which the applications are running. The connection is only a "logical" relationship made possible through Communication Manager Branch's CTI features.

Both personal desktop CTI-enabled application scenarios (controlling the phone on a single desktop) and server CTI-enabled applications (controlling/monitoring multiple phones) can be run together on the same application computer.

Avaya Aura Communication Manager Branch application enablement software

Avaya Aura Communication Manager Branch application enablement software includes:

- Core (server) software that is preinstalled on the Communication Manager Branch.

- The Avaya Application Enablement Services TAPI Telephony Service Provider, which you install on each CTI-enabled application computer. This software is also referred to as the *Avaya TSP TAPI*.
- The TSAPI client software library.

To support TAPI-based CTI-enabled applications, the TSAPI client library must be installed along with the Avaya TSP TAPI. However, if you are using only TSAPI-based applications (i.e., no TAPI applications) on the client application computer, then you can install the TSAPI client library alone.

Core software

The core software (preloaded in the Communication Manager Branch server) manages all the CTI connections between Communication Manager Branch and the CTI-enabled application computers on which the Avaya TSP TAPI and/or TSAPI software is installed. It also authenticates each application computer before establishing a CTI session to ensure that only authorized CTI User Accounts gain access to CTI services. Even though this software is preinstalled, it must be configured properly (using the Branch Device Manager) before it can be used.

Avaya CTI client software

You must install the Avaya TSP TAPI and/or TSAPI client software on each application computer on which CTI-enabled applications will be run. [Table 1](#) lists the different installation options.

Table 1: CTI client software installation options

Avaya Client Software Needed	Client Operating System	Installation Tool to Use
TAPI alone	Microsoft Windows XP (SP2)	Avaya TSP Client Software Installer (CSI)
TAPI and TSAPI	Microsoft Windows XP (SP2)	Avaya TSP CSI
TSAPI alone	Microsoft Windows XP (SP2)	Avaya TSP CSI
TSAPI alone	Microsoft Vista, 2000, and 2003 (32-bit)	Application Enablement Services TSAPI Client for Windows (refer to Installing and Configuring TSAPI Client Software)
TSAPI alone	Linux	Application Enablement Services TSAPI Client for Linux (refer to Installing and Configuring TSAPI Client Software)

Overview of Avaya Aura Communication Manager Branch application enablement

The Avaya CTI Client on each application computer establishes a CTI connection with Communication Manager Branch and exchanges service requests and event messages with the TAPI- and/or TSAPI-based CTI-enabled applications on the same computer.

Communication Manager Branch application enablement supports these TAPI versions:

- TAPI 2.1
- TAPI 2.2

Communication Manager Branch application enablement also provides a full implementation of Avaya TSAPI 4.2 including all private data versions 5 and later.

Avaya CTI client software installer

An InstallShield-based Client Software Installer (CSI) is provided to facilitate installation of both the TAPI TSP and TSAPI client library on Windows XP (SP2) based client application computers.

Note:

The Avaya TAPI TSP is only supported on Windows XP. However, the TSAPI client library can be installed on other operating systems. If you need to install only the TSAPI library on non-XP application computers then the CSI is not used. For more information on this installation procedure see [Installing and Configuring TSAPI Client Software](#) on page 37.

Application computers

Application computers must be connected to the Communication Manager Branch server over a LAN. These computers receive communications services from components of Communication Manager Branch application enablement that are installed on the Communication Manager Branch server.

Note:

Connecting client application computers to the Communication Manager Branch server over a WAN is not supported.

Client hardware requirements

Table 2: Resource requirements for application computer

Resource	Minimum Required	Comments
Dynamic Memory	512 MB	
Available Hard Disk Capacity	50 MB	
Processor Performance	550 MHz Pentium III-class	
Ethernet LAN	10 Mbps	Wired connectivity to the Communication Manager Branch System. Wireless LAN may decrease security unless properly encrypted.
Network latency	- Maximum average round trip time of 200 milliseconds (ms). - Average round trip packet delivery time as measured with ping over every one hour time period with no single spike exceeding five seconds. - No unrecovered packet loss	Avaya Services Organization is not responsible for troubleshooting a customer's LAN/WAN environment.

Limitations

Communication Manager Branch application enablement has certain limitations that affect all CTI-enabled applications. There are additional limitations and Communication Manager Branch-specific behaviors associated with TAPI and TSAPI (such as the specific services, events, and parameters that are supported in each API) that are described in this document.

Communication Manager Branch application enablement:

- Does not provide CTI support for SIP telephones.

Note:

Some Avaya telephones are capable of using either SIP or H.323 mode. Such telephones are supported when in H.323 mode.

- Does not support multi-stage dialing with TAPI or TSAPI.
- Does not support multi-address dialing strings with TAPI or TSAPI.
- Does not provide CTI support for the following DECT wireless phones:
 - WT3701
 - WT3711
- Does not restrict access to specific CTI capabilities on a per-user basis. A client application computer that is authenticated to use CTI can access all CTI telephony services and events that Communication Manager Branch application enablement provides. However, the CTI-enabled applications themselves can be designed to limit access. For example, a desktop application might require a user to enter an authorization code before the user can make a click-to-dial call. Such a requirement is managed by the application, not by Communication Manager Branch application enablement.
- Does not restrict an application from accessing any specific device or invoking any specific service. For example, there is no way to restrict users from configuring their Avaya TSP TAPI or TSAPI client software so that an application can make calls from someone else's telephone.
- Does not control VoIP phones or other local devices that exist in local PC software or are connected only to local PC interfaces. In Communication Manager Branch, CTI-enabled applications are only able to control and monitor telephony devices that are controlled by the Communication Manager Branch Feature Server (CMBFS). For example, there is a common TAPI application scenario which enables "click-to-dial" from a phone connected directly to the user's PC. This scenario is not supported by Communication Manager Branch application enablement.
- Does not support complete call-center operations, although some call center-functionality can be supported.

Capacities

Using the i40, i120, or G450 platform, you can run 50, 144, or 400 concurrent CTI-enabled application sessions, respectively. These numbers are guidelines reflecting the number of sessions and their effect on performance.

UUI Support

Communication Manager Branch 2.0 supports UUI on SIP trunks. This enables UUI information provided with the call to continue to be transported in the signalling as it is carried through SIP networks. In addition, Communication Manager Branch 2.0 includes UCID support. These features are important for supporting the transfer of call data in a centralized call center configuration, particularly when calls are transferred to and from branch office locations.

Chapter 3: Communication Manager Branch Server CTI Configuration

Although Communication Manager Branch application enablement is preinstalled on the Communication Manager Branch server, it must be configured. There are two configuration areas:

1. [CTI User Account authentication](#) (required)
2. [Creating CTI Extensions](#) (optional) - These are special extensions with no associated physical hardware that can be used for particular CTI-enabled application purposes. If your applications do not specifically require CTI Extensions then there is no need to create or configure them.

CTI User Account authentication

When a CTI-enabled application initiates a CTI session with Communication Manager Branch application enablement, the application provides credentials (a user ID and password) to a particular CTI User account. Communication Manager Branch application enablement uses this ID and password to ensure that the application is valid and permitted to use CTI.

Note:

Avaya recommends that you administer CTI User Account IDs and passwords to each CTI enabled application before installing Avaya TAPI TSP software on application computers. This will allow each installation to be tested immediately for correct operation.



Tip:

A "CTI User Account" administered on the Communication Manager Branch server is similar to an "account" that can be shared by one or more CTI-enabled applications when starting a CTI session.

Communication Manager Branch uses one of the following methods to authenticate the credentials (CTI User Account ID and Password) supplied by a client CTI-enabled application when it attempts to start a CTI session.

- Local CTI user account authentication. When this method is used, Communication Manager Branch application enablement uses an internal file of administered CTI user account IDs and passwords that is stored on the Communication Manager Branch system. This method is the default method for CTI user account authentication. See [Local CTI user account authentication](#) on page 18.

- Windows Active Directory Services authentication. When this method is used, Communication Manager Branch application enablement uses an external, customer-provided Active Directory server to perform the authentication. You must set up the Active Directory and configure Communication Manager Branch application enablement to connect to the Active Directory server. See [Microsoft Active Directory Services for user account authentication](#) on page 19.

Local CTI user account authentication

When local user account authentication is used, CTI enabled applications are authenticated against the accounts in the internal file that are created through the local administration on the Communication Manager Branch system. For security, only authenticated CTI-enabled applications are granted access to Communication Manager Branch CTI services.

A maximum of five CTI user accounts (that can be shared amongst all application computers) are configurable in the main CTI configuration screen.

To use local CTI user account authentication, use the CTI Configuration screen ([Figure 2](#)) in Branch Device Manager to create up to five CTI user accounts. To go to this screen, from the **Navigation** pane on the main menu click **Managed Objects > Resources > CTI Configuration**.

Figure 2: CTI Configuration Screen

The screenshot shows the Avaya Communication Manager Branch Edition Device Manager interface. The left sidebar contains a 'Managed Objects' tree with 'Configuration' expanded, showing 'Users', 'Group Communication', 'Call Handling', 'Automated Attendant Service', 'Public Networking', 'Resources', 'System Parameters', and 'Platform'. The main content area is titled 'CTI Configuration' and includes buttons for 'Apply Changes', 'Add User', and 'Remove User'. It features sections for 'Authentication Method' (with radio buttons for 'Active Directory Server' and 'Local Users'), 'Active Directory Server Configuration' (with fields for 'Server Address', 'Port', 'Realm', and 'Keytab File Status'), 'Tserver Configuration' (with fields for 'Tserver Port' and 'Tserver Switch Port'), 'Update Keytab File' (with a 'Browse...' button), and 'Local CTI Users' (a table with columns for 'Select', 'Username', 'Password', and 'Confirm').

Select	Username	Password	Confirm
☐	dan	*****	*****
☐	ctiuser	*****	*****

This screen allows you to add up to five local CTI user accounts. To add a CTI user account:

1. Click the **Add User** button once for each user account you want to add.

The bottom of the screen will display fields for adding one new user each time you click the button. For example, if you click the button twice, fields for adding two new users will be displayed, as shown in [Figure 2](#) above.

2. Under **Local CTI Users**, check the **Select** button and fill in the following fields:

- **User** - the User ID.
- **Password** - the password. The password must contain between 8 - 32 alphanumeric characters, with the first character being a letter.
- **Confirm** - retype the password.

3. Click the **Apply Changes** button.

For more information, see **CTI** in the Branch Device Manager Online Help.

Microsoft Active Directory Services for user account authentication

When you use Microsoft Active Directory Services (ADS) for CTI user account authentication, credentials provided by CTI-enabled applications are authenticated against an external, customer-provided Active Directory Domain Controller database. To use ADS for authentication, you must configure Communication Manager Branch application enablement as an Active Directory Kerberos client.



Important:

When using Active Directory services for authentication, the clocks in the Communication Manager Branch system and Active Directory server must always be maintained within two minutes of each other. The clock in the Communication Manager Branch system is maintained on the **Date and Time** screen in the Branch Device Manager. To go to this screen, from the navigation pane on the main menu click **Platform > Date and Time**. For more information, go to the Branch Device Manager Online Help.

To configure the system to use ADS for authentication, perform the following procedures:

1. Create an account on the Domain Controller.
2. Generate a keytab file for the account on the Domain Controller.
3. Install the keytab file and enable Active Directory authentication in Branch Device Manager.

Creating a CTI User Account on the Domain Controller

On the Domain Controller, follow this procedure to create a CTI User account where the Communication Manager Branch application enablement is designated as an Active Directory user.

Note:

To perform this procedure, you must have administrator permissions on the Windows Domain Controller.

1. From your desktop, select **Start > Settings** and click **Control Panel**.
2. From **Control Panel**, double-click **Administrative Tools**.
3. From **Administrative Tools**, double-click **Active Directory Users and Computers**.
4. In the left pane, click **Users** to display the list of users. Move your cursor to the right pane and right-click. From the right-click menu, select **New > User**.
Windows displays the **New Object - User** dialog box.
5. Complete the **New Object - User** dialog box as follows:
 - a. In the **First Name** field, type a user name. For example, use the host name of the Communication Manager Branch system.
 - b. Leave the **Initials** and the **Last Name** fields blank.
 - c. In the **Full Name** field, type exactly what you typed in as a user name in the **First Name** field.
 - d. Complete the two-part **User logon name** field, as follows:
 - In the first field, type the user name.
 - In the next field, type the address (for example **@dcserver1.xyz.com**) of the Domain Controller.
6. Click **Next**.
Windows displays the **New Object - User** dialog box.
7. Complete the **New Object - User** dialog box as follows:
 - a. In the **Password** field, type a password you have chosen.
 - b. In the **Confirm password** field, retype the password.
8. Click **Next** and **Finish** on the remaining screens to complete the procedure.

Generating a keytab file for the account on the Domain Controller

To perform this procedure, you must be "logged" on the Windows Domain Controller with administrator permissions. After you create the Communication Manager Branch application enablement account on the Windows Domain Controller, generate a keytab file for the account.

Note:

This example uses a CTI user account name to generate a keytab file that is called **username.keytab**. This file will be created in the current directory. Make a note of this location as you will need this location during the installation procedure.

On the Domain Controller, start a command prompt and type:

```
Ktpass -princ host/hostname@NT-DNS-REALM-NAME -mapuser username -pass  
password -out username.keytab
```

where:

host is the Communication Manager Branch host name

hostname is the Communication Manager Branch's DNS name

NT-RDNS-REALM-NAME is the Active Directory domain name you are using to authenticate

username is the username that you created in Active Directory Services

password is the password for the account.

Installing the keytab file and enabling Active Directory authentication in Branch Device Manager

To install the keytab file and enable Active Directory authentication, use the CTI Configuration screen (Figure 3) in Branch Device Manager. To go to this screen, from the **Navigation** pane on the main menu click **Managed Objects > Resources > CTI Configuration**.

Figure 3: - Branch Device Manager CTI Configuration screen

The screenshot displays the Avaya Communication Manager Branch Edition Device Manager interface. The left-hand navigation pane shows the 'Managed Objects' tree with 'CTI Configuration' selected under 'Resources'. The main content area is titled 'CTI Configuration' and includes the following sections:

- Buttons:** 'Apply Changes' (green), 'Add User' (blue), and 'Remove User' (red).
- Authentication Method:** Radio buttons for 'Active Directory Server' (selected) and 'Local Users'.
- Tserver Configuration:** Fields for 'Tserver Port' (450) and 'Tserver Switch Port' (1050).
- Active Directory Server Configuration:**
 - Server Address:** Text input field.
 - Port:** Text input field with '88' entered.
 - Realm:** Text input field.
 - Keytab File Status:** Displayed as 'Uninstalled'.
- Update Keytab File:** A section with a text input field, a 'Browse...' button, and an 'Update!' button.
- Local CTI Users:** A table with columns: 'Select', 'Username', 'Password', and 'Confirm'.

1. Under **Authentication Method**, select the **Active Directory Server** radio button.
2. Fill in the following fields:
 - **Server Address** - the IP address of the Active Directory server.
 - **Realm** - the Active Directory domain name you are using to authenticate. This should be the same Realm you used when running the command to generate the keytab file. See [Generating a keytab file for the account on the Domain Controller](#) on page 20

Note:

The **Port** field is populated by a default value.

3. In the **Update Keytab File** area browse to the location of your generated keytab file.

4. Click **Update**.

The keytab file that was generated on the Domain Controller is loaded into Communication Manager Branch.

5. Click the **Apply Changes** button.

For more information, see the **CTI** page in the Branch Device Manager Online Help.

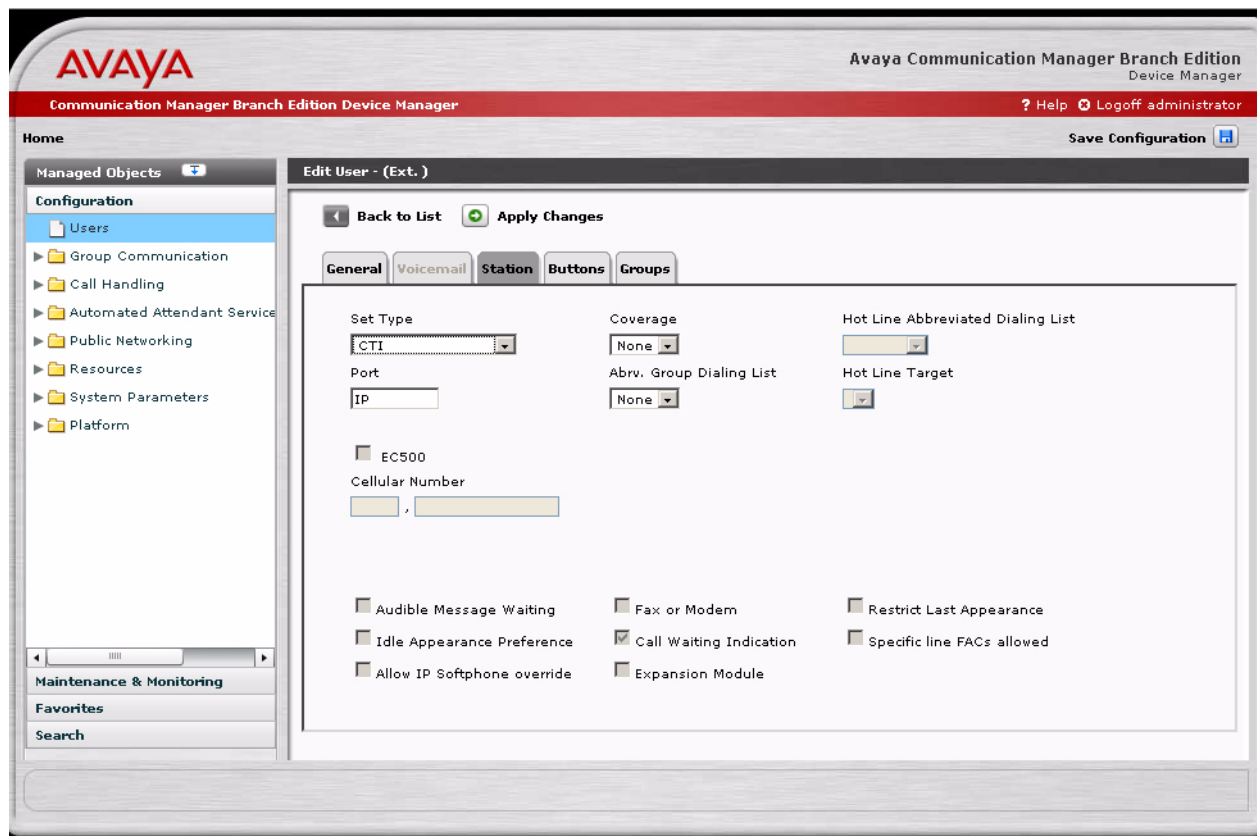
Creating CTI Extensions

Each CTI Extension acts like a standard phone in almost all regards, including counting in the total number of extensions on your Communication Manager Branch. A CTI Extension is a special station type in that it has an extension number, but there is no physical hardware associated with the number. A "normal" station type is an actual phone that sits on a desktop. The CTI Extension station type exists only in the Communication Manager Branch server software. However, under CTI control this "virtual phone" can interact with calls like an actual phone. Calls can be delivered to the CTI Extension, be put on hold, transferred, and so forth. This ability to act like an actual phone without the accompanying expense or wiring makes CTI Extension type stations very useful for certain CTI purposes. If you have a CTI-enabled application that requires CTI Extensions, then you must create them using the following administration procedure on the Branch Device Manager.

Just like physical phones, CTI Extensions can have Call Appearance and Line Appearance buttons assigned to them. However, no other buttons can be configured.

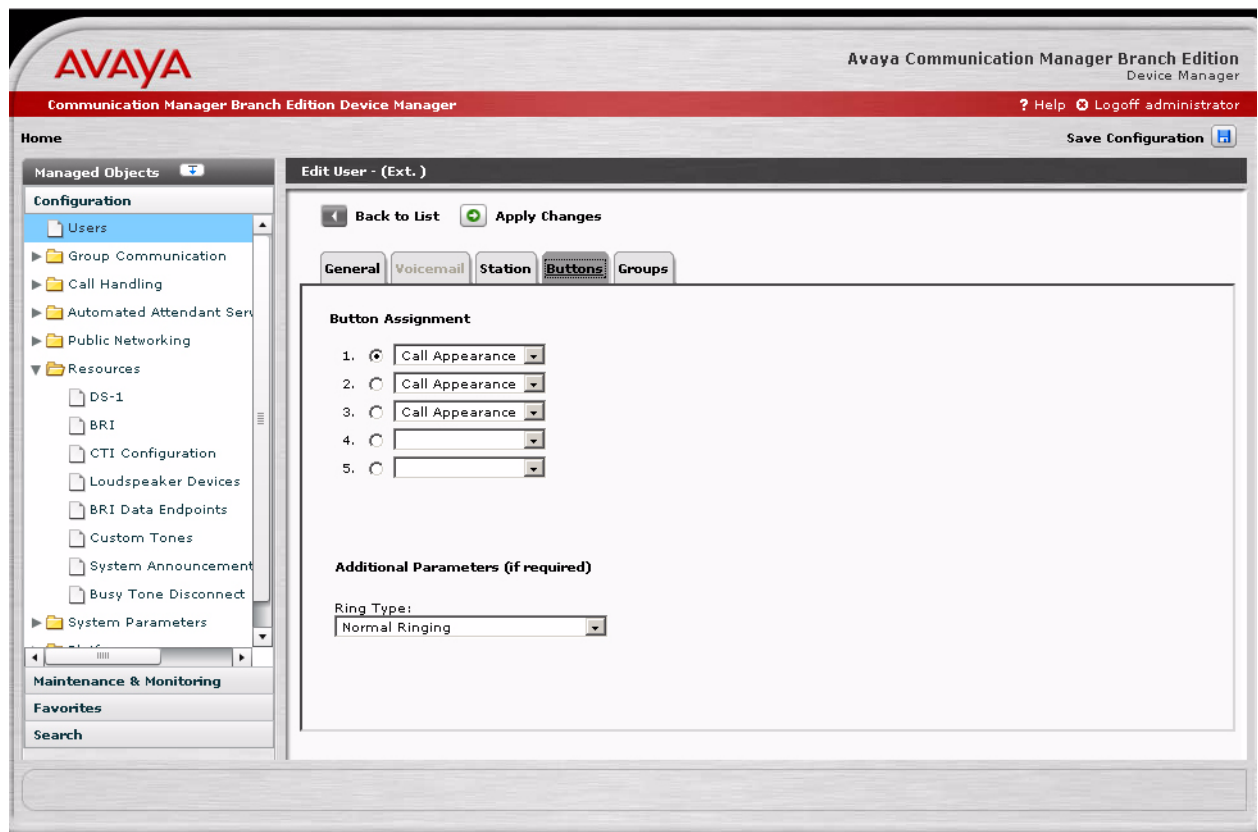
To create a CTI Extension, use the **Edit User - (Ext) Station Screen** ([Figure 4](#)) in Branch Device Manager. To go to this screen, from the Navigation pane on the main menu click **Managed Objects > Users**. The following page appears:

Figure 4: Edit User Station Screen



1. Click the **Stations** tab.
2. From the Set Type drop down menu, select CTI.
3. Click the **Buttons** Tab. The Button Assignment screen appears.

Figure 5: Button Assignment screen



4. Assign up to five Call or Line Appearances (default: three Call Appearance buttons).

Chapter 4: Installing and Configuring the Avaya CTI Software on each Application Computer

Introduction



Tip:

Although not mandatory, it is recommended that you complete all CTI-related administration and configuration on the Communication Manager Branch server before installing Avaya CTI software on client application computers. This allows testing and verifying the client installations immediately, using the provided installation test application (the Installation Test Dialer). On the client installation, the Installation Test Dialer establishes a CTI connection to the Communication Manager Branch server and attempts to place a call. A placed call means that the installation was successful.

The CTI client software consists of three components:

1. A TAPI Telephony Service Provider (TSP)
 2. An Installation Test Dialer
 3. TSAPI client software
- To enable TAPI-based applications on a client application computer, install all three components.
 - To enable only TSAPI-based applications, install the TSAPI client software alone.

The Avaya Client Software Installer (CSI) installs any or all of these components on a client application.

Note:

The Avaya TAPI TSP is only supported on Windows XP. However, the TSAPI client library can be installed on other operating systems. If you need to install only the TSAPI library on non-XP application computers then the CSI is not used. For more information on this installation procedure see [Installing and Configuring TSAPI Client Software](#) on page 37

To set up an application computer you must:

1. Install the Avaya TAPI TSP.
2. Configure the Avaya TAPI TSP.

Installing the Avaya TAPI TSP



Important:

The following information describes how to install the TAPI TSP for TAPI software. If you are installing only the TSAPI software, refer to [Installing and Configuring TSAPI Client Software](#).

Preinstallation tasks

- Verify:
 - that each application computer is using Windows XP Professional operating system with Service Pack 2 or later.
 - that Communication Manager Branch application enablement is configured and running on the Communication Manager Branch server. See [Monitoring CTI status](#) on page 46 for information on how to verify this.
 - that the application computer meets the minimum hardware requirements. See [Table 2: Resource requirements for application computer](#) on page 13.
- Determine the IP address of the Avaya Aura Communication Manager Branch application enablement server.
 - Navigate to **Managed Objects > Maintenance & Monitoring > System Summary** and locate the "IP Address" field.
- Configure your telephony dialing rules using the Phone and Modem Options of the Control Panel, per Microsoft documentation.
- Obtain the Avaya TAPI TSP software from the Avaya DevConnect site (<http://www.avaya.com/devconnect>). To install the software you will need to download the zipped file to an available temporary directory on the installation computer and extract the files into this same directory. Optionally, you may now want to copy these files to an installation CD-ROM that you can use to install onto other application computers, rather than downloading the zipped file again on each computer.

Note:

Access to DevConnect resources requires a valid DevConnect Program Login ID and password. For more information on the Avaya DevConnect Program, please refer to [Appendix D: About the Avaya DevConnect Program](#) on page 119.

- Determine whether you will use encrypted (secure) or unencrypted (non-secure) connections:

Communication Manager BranchR2.0 offers both encrypted and unencrypted connections for the LAN connection between client application computers and the Communication

Manager Branch server. This enables application computers using earlier versions of client software to continue to connect to the Communication Manager Branch2.0 server using unencrypted connections, while application computers using Release 1.2 client software can optionally connect on an encrypted connection. During Avaya TAPI TSP or TSAPI application configuration you will be able to choose the desired type of connection. You can change from one setting to the other at any time.

Note:

Even unencrypted connections are encoded in a binary format that is not readable without special processing.

Installation procedure

Note:

TSAPI is an additional Avaya API for application enablement. The Avaya TAPI TSP uses some Avaya TSAPI components that are installed on each application computer. Because of this, TSAPI is referenced in some messages during the installation.

1. Start the InstallShield Wizard by running the **setup.exe** application that is at the top of the directory structure of the extracted files that you obtained from the Avaya DevConnect site.
The InstallShield Wizard starts. Click **Next** to continue.
2. Select **Complete** and click **Next**.
3. Specify a destination folder and click **Next**.
4. Type the IP address of the Communication Manager Branch application enablement server that you determined during the preinstallation tasks. Click **Next**.
5. You may receive a message about the installation of the TSAPI client. Click **OK** to proceed.
6. Review your settings and click **Next** to start the installation.

A status bar shows the progress of the installation.

7. Select whether to restart the application computer now or later and click **Finish**.

We recommend that you select to restart later.

Note:

You must restart the application computer after configuring the Avaya TAPI TSP. To avoid having to restart the computer twice, Avaya recommends that you complete the configuration, see [Configuring the Avaya TAPI TSP](#) on page 30, and then restart the computer.

Configuring the Avaya TAPI TSP

Before you use the Avaya TAPI TSP with a TAPI-based application, you must configure the TSP on the application computer.

Note:

Make sure you have determined what the CTI user ID and password will be for this particular application computer. Also, the CTI user ID and password must be administered in Communication Manager Branch application enablement before you will be able to test the installation.

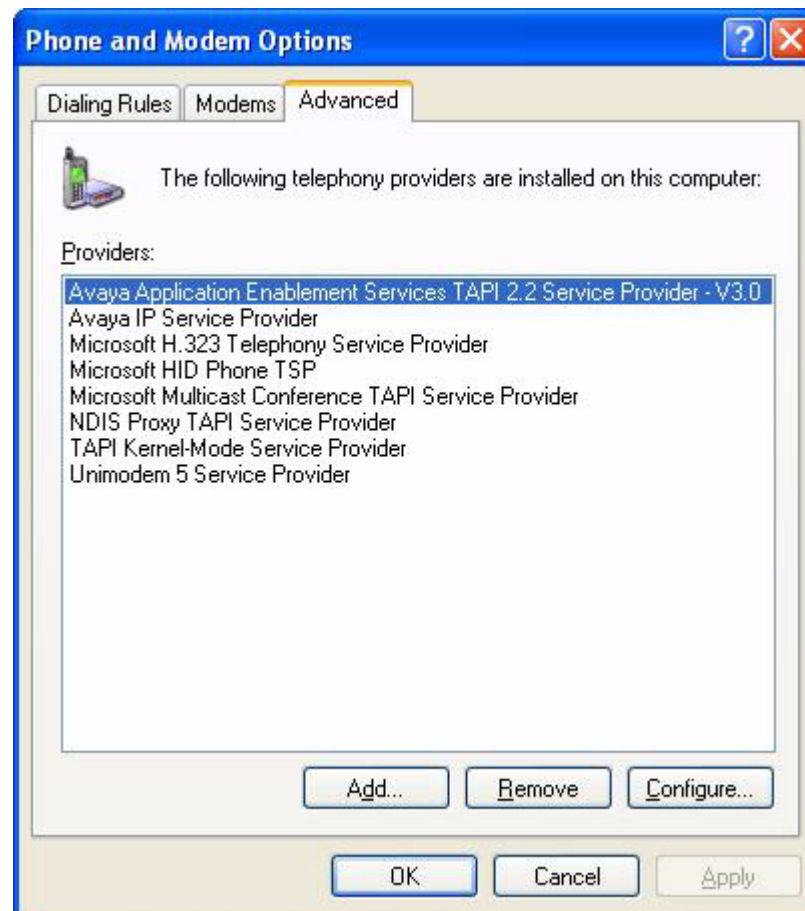
To configure the Avaya TAPI TSP:

1. From your desktop, select **Start > Settings** and click **Control Panel**.
2. From **Control Panel**, double-click **Phone and Modem options**.

Note:

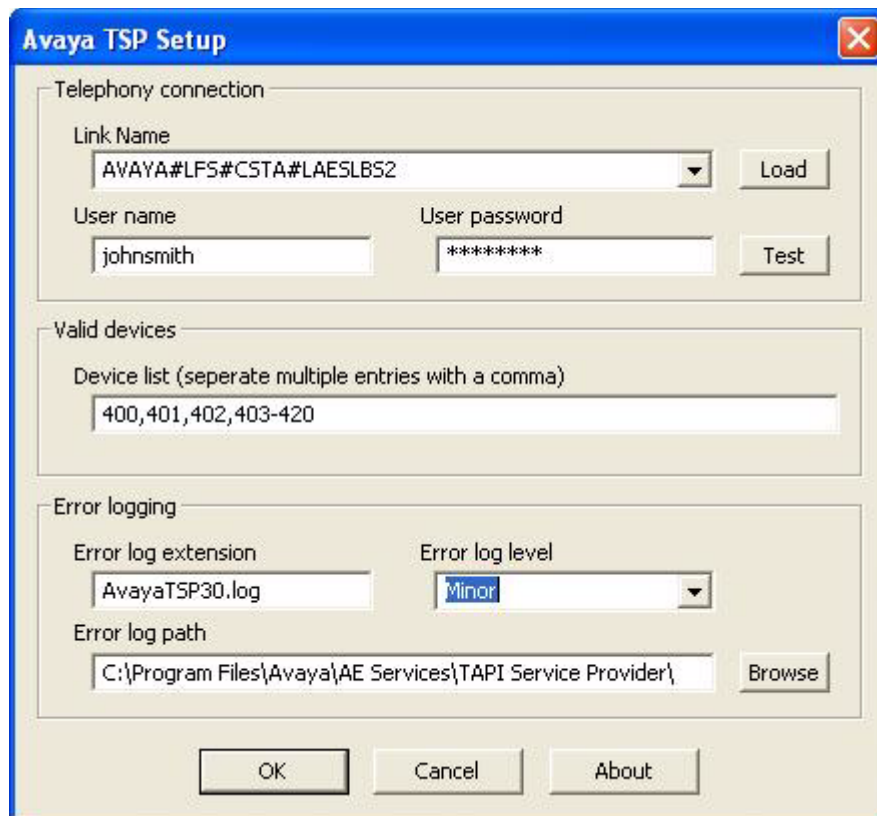
If telephony information is not yet set up for this computer, the system asks you for an area code and location. Provide this information and click **OK**. The system then displays the Phone and Modem options screen.

3. Click the **Advanced** tab.
4. From the list of available telephony providers, select **Avaya Application Enablement Services TAPI 2.2 Service Provider V3.0** and click the **Configure** button.

Figure 6: Phone and Modem options

The computer displays the Avaya TSP Setup dialog box.

Figure 7: Avaya TSP Setup dialog box



5. Complete the Avaya TSP Setup dialog box as follows:

- Choose a Link Name with the format AVAYA#LFS#**CSTA**#*systemname* to select an unencrypted (non-secure) connection (where ***systemname*** is the name of the Communication Manager Branch system). Choose AVAYA#LFS#**CSTA-S**#*systemname* to select an encrypted (secure) connection. Both of these selections will always be available.
- Click Load.
- In the **User name** text box, type the CTI user ID as administered on Communication Manager Branch.
- In the **User Password** field, type the password that is associated with the CTI user ID.

Note:

To test the installation and configuration of the Avaya TAPI TSP, the CTI user ID and password will have to be administered on Communication Manager Branch or in Active Directory Services as a valid user. See [Local CTI user account authentication](#) on page 18 and [Microsoft Active Directory Services for user account authentication](#) on page 19

- In the **Device list** field, enter the extension number (Device Number (DN)) of all phones that will be monitored or controlled by the CTI-enabled applications on this application computer.

Note:

You must enter the DNs of all extensions that **ANY** application on this computer application might use. For example, if two TAPI-based applications are to be installed on this computer and one controls only the deskphone and the other controls or monitors all the phones in the branch, enter all the DNs in this text box.

- Entering DNs:
 - To enter a single DN, type the number.
 - Separate multiple DNs with a comma (,).
 - Enter ranges of DNs using a "dash". For example, to include all the following DNs 1234, 1235, 1236, 1237, type 1234-1237.
 - Enter both multiple DNs and ranges, separating them with a comma. For example: 2767, 2014, 1234-1237, 7231, 4000-4199
- If you want to change the default path for the storage location of the error log, click **Browse** and select a storage location for the error log.
- In the **Error log level** field, accept the default error log level of **ERROR_MINOR** (this should be applicable for most situations), or select another:

ERROR_FATAL. Logs fatal error messages.

ERROR_MAJOR. Logs fatal and major error messages.

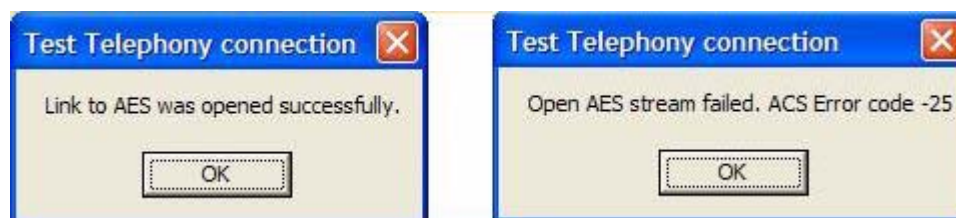
ERROR_MINOR. Logs fatal, major and minor error messages.

ERROR_TRACE. Logs fatal, major and minor errors and trace information.

To save your configuration settings, click **OK**.

6. To test the configuration, click the **Test** button. If you have configured the system correctly (i.e. fields have correct information, the Communication Manager Branch hardware is set up and configured) you will receive a success message. If there is a problem, you will receive an error message. [Figure 8](#) shows a sample success message and an error message (the error message will vary, depending on the error).

Figure 8: Test Messages



Installing and Configuring the Avaya CTI Software on each Application Computer

7. You will receive a warning message that your changes will not take affect until you have restarted the computer. Click **OK** to proceed.

Note:

With the Phone and Modem options, you can change the Avaya TAPI TSP configurations, even when telephony operations (sessions) are in progress. These changes, however, do not take effect until you shut down and restart the computer.

8. Restart the application computer.

Running the Avaya TAPI TSP installation test

You can test the installation after:

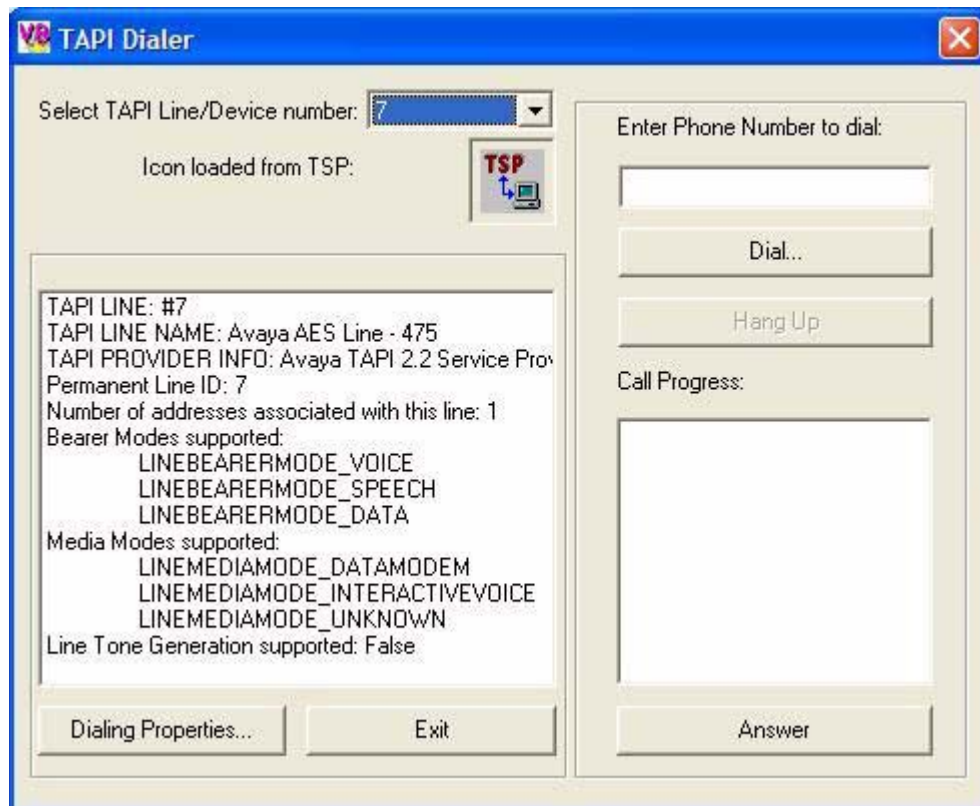
- the Avaya TAPI TSP has been configured for the application computer.
- the application computer has been restarted.
- the CTI user ID's and passwords have been administered on Communication Manager Branch.

To test the installation:

- Navigate to **Start > All Programs > Avaya AE Services > TAPI Service Provider** and select the **TAPI Dialer** application.

The system displays the TAPI Dialer dialog box.

Figure 9: TAPI Dialer dialog box



- Select the **TAPI Line/Device number** that corresponds to the **TAPI LINE NAME: Avaya AES Line**.

- In the **Enter Phone Number to Dial** field, enter a known working local extension number other than your own.
- Click the **Dial** button.

Call progress messages will appear in the space provided.

Removing the Avaya TAPI TSP

To remove completely the Avaya TAPI TSP, you must remove the following separate components in the order shown.

- Avaya Application Enablement Services TAPI Service Provider
- Avaya Application Enablement Services TSAPI Client

To remove these components:

1. From your desktop, select **Start > Settings** and click **Control Panel**.
2. From **Control Panel**, double-click **Add or Remove Programs**.
3. From the list of currently installed programs, select **Avaya Application Enablement Services TAPI Service Provider**. Click **Change/Remove**.
4. Select **Remove** to remove all installed features and click **Next**.
5. The system provides messages that confirm that the selected software was removed and maintenance is complete. Click **OK** and **Finish** when requested.

The system displays the **Add or Remove Programs** screen.

Remove the following component in a similar way:

- Avaya Application Enablement Services TSAPI Client

Reinstalling the Avaya TAPI TSP

To reinstall the Avaya TAPI TSP after you have installed and removed at least once:

1. Start the InstallShield Wizard by running the **setup.exe** application that is at the top of the directory structure.

The Modify, repair, or remove screen appears.

Figure 10: Modify, repair, or remove screen

Select **Remove** and click **Next**.

Note:

This removal process is necessary to ensure that all components are properly reinstalled

2. The system provides messages that confirm that the selected software was removed and maintenance is complete. Click **OK** and **Finish** when requested.
3. Restart the installation process. See [Installing the Avaya TAPI TSP](#) on page 28.

Installing and Configuring TSAPI Client Software

By installing the Avaya TAPI for TSP, the TSAPI client is installed automatically. Therefore, no further installation is required. Use the information provided in the following sections if your CTI applications require the TSAPI client only and you have not already installed the Avaya TAPI TSP.

Preinstallation Tasks

- Verify:
 - that Communication Manager Branch application enablement is configured and running on the Communication Manager Branch server. See [Monitoring CTI status](#) on page 46 for more information.
 - that the application computer meets the minimum hardware requirements. See [Table 2: Resource requirements for application computer](#) [Resource requirements for application computer](#) on page 13.
- Determine:
 - the operating system used on the application computer(s) onto which you will be installing the Avaya TSAPI client software
 - the IP address of the Communication Manager Branch application enablement server

If you do not have the Host name or IP address of the Communication Manager Branch application enablement server, you can still install the client.

Follow the installation procedure (Procedure to install the TSAPI Windows client on page 27), and at Step 7 complete the TCP/IP Name Server Configuration dialog box as follows.

 1. In the Host Name or IP Address field, type a place holder, such as myserver or 1.2.3.4 and accept the default port number. The setup program issues a Warning dialog when you click Add to List or Next, but it lets you continue with the installation.
 2. After installation, when you get a valid host name or IP address, you will need to edit the tslib.ini file and add the appropriate host name or IP address for the Communication Manager Branch application enablement server.
- From the Avaya DevConnect site (<http://www.avaya.com/devconnect>), download the Avaya TSAPI client software appropriate for the operating system(s) on the target application computer(s).
 - Download the zip file to a temporary directory and extract the files to the same directory.
 - If the installation process is to be repeated, copy the installation files to a CD.

Note:

Access to DevConnect resources requires a valid DevConnect Program Login ID and password. For more information on the Avaya DevConnect Program, please refer to [Appendix D: About the Avaya DevConnect Program](#) on page 119.

Installing and Configuring TSAPI Client Software on Linux Operating Systems

Follow this procedure to install the TSAPI Linux client.

1. Log in to the client computer as root.
2. Follow Step **a** for a CD-ROM based installation. Follow Step **b** for a Web-based installation.
 - a. Insert the Application Enablement Services 4.1 TSAPI SDK CD-ROM into your computer's CD-ROM drive. (The TSAPI SDK CD-ROM includes the client and the SDK.)
 - Type `mount /mnt/cdrom/` to mount the file system.
 - Type `cd /mnt/cdrom/client/Linux` to change to the directory containing the TSAPI Linux client .rpm file (for example, `tsapi-client-linux-4.1-x.i386.rpm`). Continue with Step 3.
 - b. Browse to the Avaya DevConnect Web site (<http://www.avaya.com/devconnect>) and log in.

Note:

The layout of the DevConnect Web site is subject to change. The following steps are based on the current layout of the DevConnect Web site.

- From the "Welcome" page, in the menu on the left, click **Avaya Product Information/ Documentation/SDKs**.
 - From the "Avaya Product Information" page, under "Premium DevConnect Content: SDKs & APIs," click **SDK & API Index, Sorted by Avaya Platform**.
 - From the "Premium DevConnect Content: SDK & API Index" page, under "Developer Tools by Avaya Product or Solution," and "Application Enablement Services (AE Services) for Communication Manager," click **Application Enablement Services (AE Services) SDKs**.
 - From the "Application Enablement Services (AE Services)" page, under "AE Services Technical Resources" and "AE Services SDKs & Client Software," click **AE Services TSAPI SDK**.
 - From the "AE Services: TSAPI SDK" page, under "AE Services TSAPI Technical Resources", which is followed by "AE Services TSAPI SDK, Client Software and Documentation," click the link.
 - From the "AE Services Release 4.1 Contents" page, under "AE Services 4.1 Downloads," click **AE Services 4.1 TSAPI Client [Linux]**, and save the file (`tsapi-client-linux-4.1-x.i386.rpm`) to your computer.
 - Go to the directory where you downloaded the client (`tsapi-client-linux-4.1-x.i386.rpm`), and continue with Step 3.
3. Install the TSAPI Linux client using the rpm command.

For example: `rpm -i tsapi-client-linux-4.1-x.i386.rpm`

4. Use the **rpm** command to verify that the Linux client is installed.

For example: `rpm -q tsapi-client-linux`

5. The system will display the file name of the Linux client if it is installed;

For example: `tsapi-client-linux-4.1-x`

- 6.

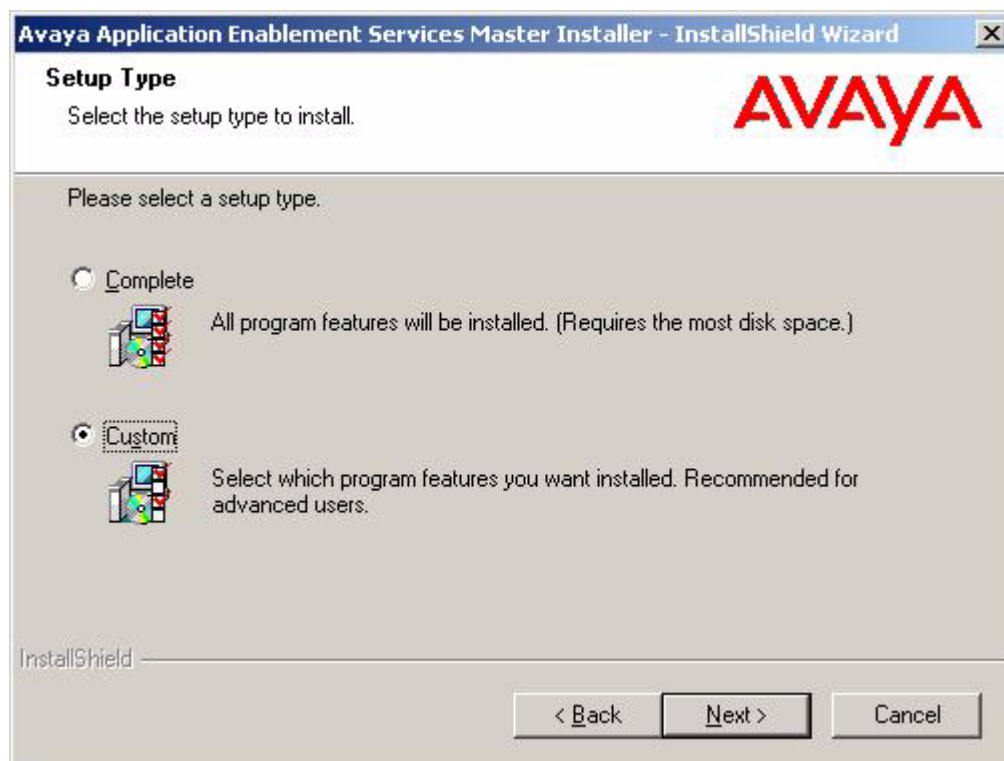
Installing and Configuring TSAPI Client Software on Windows XP

This procedure describes how to install only the TSAPI Client software on a Microsoft XP operating system

Normally, when you use the CSI to install the TAPI TSP on a Windows XP OS, the TSAPI client is automatically installed as well (the Avaya TAPI TSP cannot operate without TSAPI). However, TSAPI itself can be installed as a standalone on a Windows client.

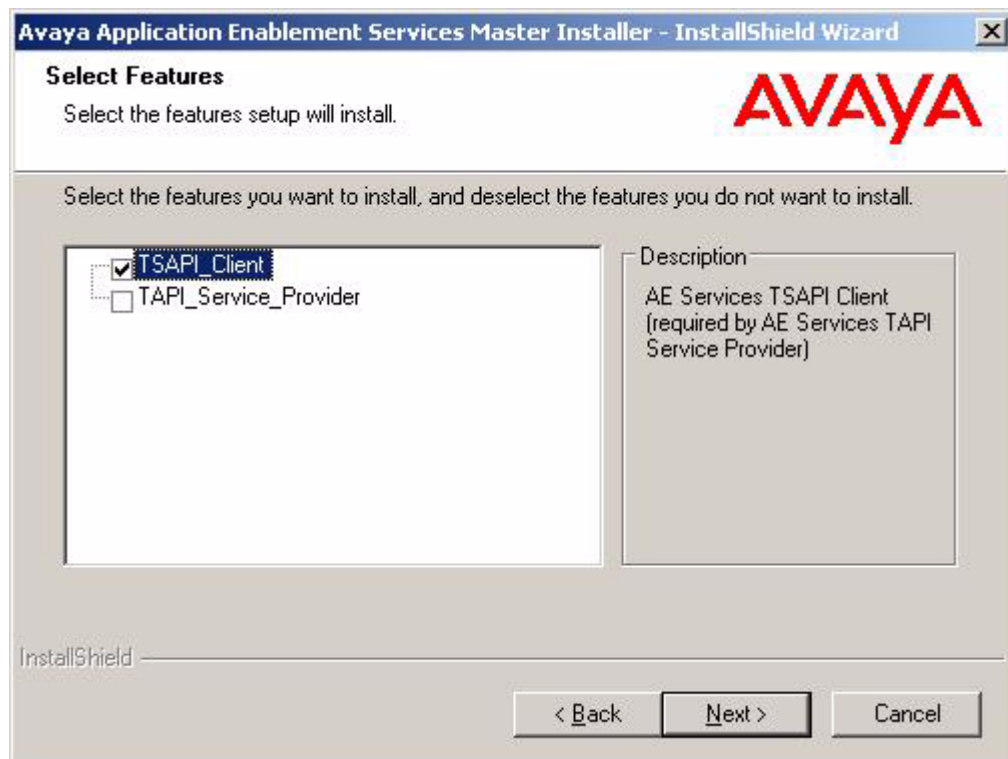
To install TSAPI:

Figure 11: Custom Install TSAPI only



1. Start the InstallShield Wizard by running the setup.exe application that is at the top of the directory structure of the extracted files that you obtained from the Avaya DevConnect site.
The InstallShield Wizard starts.
2. Click **Next** to continue.
3. Select the Custom setup type (complete or custom) to install and click **Next**.
4. Specify a destination folder and click **Next**.
5. Check the box for **TSAPI Client** and leave TAPI Service Provider blank.

Figure 12: Selecting TSAPI Client



6. Type the domain name or the IP address of the Communication Manager Branch application enablement server that you determined during the preinstallation tasks.
7. Click **Next**.
8. Review your settings and click **Next** to start the installation.
A status bar shows you the progress of the installation.
9. Click **Finish**.

Installing and Configuring TSAPI Client Software on Other Systems

This procedure describes how to install only the TSAPI Client software on a Microsoft Vista operating system

Use the following procedure to install the TSAPI Windows client.

1. Log on to your computer as a user with administrator-equivalent permissions.
2. Follow Step **a** for a CD-ROM based installation. Follow Step **b** for a Web-based installation.
 - a. Insert the Application Enablement Services 4.1 TSAPI SDK CD-ROM into your computer's CD-ROM drive. (The TSAPI SDK CD-ROM includes the client and the SDK.) Browse to **D:\AES4.1\TSAPI\CLIENT\WINDOWS**, and locate the TSAPI client file (for example **tsapi-client-win32-4.1-x.zip**). Extract the files to a directory on your computer, (for example, **C:\temp\AES4.1\TSAPI\CLIENT\tsapi-client-win32-4.1-x**), and locate the Setup.exe file. Continue with Step 3.
 - b. Browse to the Avaya DevConnect Web site (<http://www.avaya.com/devconnect>) and log in.

Note:

The layout of the DevConnect Web site is subject to change. The following steps are based on the current layout of the DevConnect Web site.

- From the "Welcome" page, in the menu on the left, click **Avaya Product Information/Documentation/SDKs**.
- From the "Avaya Product Information" page, under "Premium DevConnect Content: SDKs & APIs," click **SDK & API Index, Sorted by Avaya Platform**.
- From the "Premium DevConnect Content: SDK & API Index" page, under "Developer Tools by Avaya Product or Solution" and "Application Enablement Services (AE Services) for Communication Manager," click **Application Enablement Services (AE Services) SDKs**.
- From the "Application Enablement Services (AE Services)" page, under "AE Services Technical Resources" and "AE Services SDKs & Client Software," click **AE Services TSAPI SDK**.
- From the "AE Services: TSAPI SDK" page, under "AE Services TSAPI Technical Resources", which is followed by "AE Services TSAPI SDK, Client Software and Documentation," click the link.
- From the "AE Services Release 4.1 Contents" page, under "AE Services 4.1 Downloads," click **AE Services 4.1 TSAPI Client [Win32]**, and save the file (tsapi-client-win32-4.1-x.zip) to your computer.
- Go to the directory where you downloaded the client (tsapi-client-win32-4.1-x.zip). Extract the files, and locate the Setup.exe file. Continue with Step 3.

3. Double-click **Setup.exe**.

Setup displays the splash screen, which is followed by the Welcome dialog box.

4. Click **Next**.

Setup searches for any previously installed files.

- If Setup detects the Avaya CT TS Win32 client, it issues the following warning: "The Avaya CT Win 32 Client needs to be removed before the installation can continue." When you click OK, the installation program exits. For information about removing the Avaya CT client, see Procedure to remove the TSAPI Windows client on page 32.
- If Setup finds files from a previous version of the TSAPI client, it displays a dialog box labeled as follows: "Files from previous version of TSAPI Client were found." The text area of the dialog box contains a list of the files. These files will not conflict with your installation of the TSAPI Client, and you can decide whether to keep them or not. Click **Next** to continue with the installation.

After completing the file search, Setup displays the Choose Destination Location dialog box.

5. Click **Next** to accept the default destination folder: C:\Program Files\Avaya\AE Services\TSAPI Client.

Setup displays the **TCP/IP Name Server Configuration** dialog box.

6. Complete the **TCP/IP Name Server Configuration** dialog box.

The information you specify in this dialog box is saved in the tslib.ini file. If you do not have this information, see If you do not have the host name or IP address on page 26.

- a. In the Host Name or IP Address field type a valid host name or IP address of the Communication Manager Branch application enablement server, for example:

192.168.123.44 (IP address)

doaeserver1 or doaeserver.company.com (host name)

- b. In the TCP Port field, accept the default 450. If your installation uses more than one Communication Manager Branch application enablement server, click **Add to List**.

You can repeat substeps a and b to add multiple host names or IP addresses to the Configured Telephony Servers list box.

- c. Click Next.

Setup displays the **Setup Status** dialog box as it installs files, and then displays the Installation Complete dialog box.

7. From the **Installation Complete** dialog box, click **Finish**.

Setup exits and Windows displays your computer's desktop.

Chapter 5: Managing Avaya Aura Communication Manager Branch application enablement

Logging CTI errors and exceptions

Any errors and exceptions that the Avaya TAPI TSP detects are recorded in a specified log file that is stored on each application computer. You specify the location of this log file in [Configuring the Avaya TAPI TSP](#) on page 30. Communication Manager Branch application enablement does not send TAPI log files to centralized storage.

TSAPI errors and exceptions are logged only on the Communication Manager Branch server - not on client application computers.

Adding and deleting local CTI user accounts for authentication

Communication Manager Branch application enablement supports up to five unique CTI user accounts.

To add or delete local CTI user accounts, use the CTI Configuration screen in the Branch Device Manager. To go to this screen, from the **Navigation** pane on the main menu, click **Managed Objects > Configuration > Resources > CTI Configuration**.

For instructions for adding CTI user accounts on this screen, see [Local CTI user account authentication](#) on page 18.

To delete a CTI user account on this screen, click the **Remove User** button.

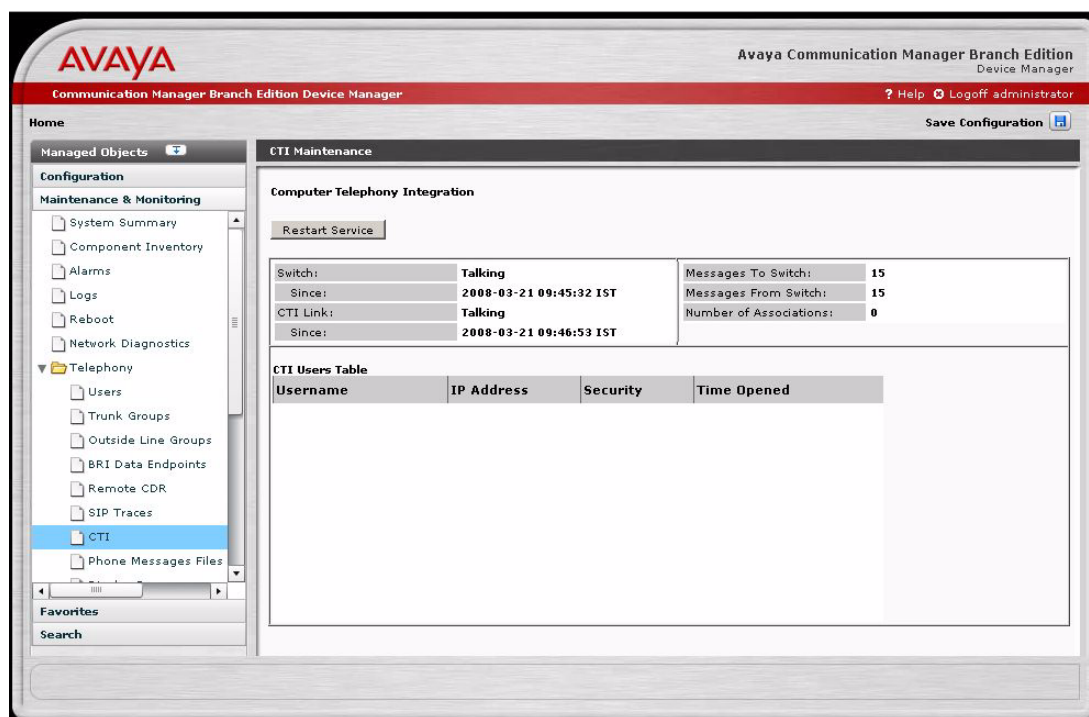
Note:

While you can edit the password, the user name can not be edited. To change a user name, you must delete that CTI user and then add the CTI user with a new user name.

Monitoring CTI status

The CTI Configuration screen in the Branch Device Manager ([Figure 13](#)) provides status and maintenance for Communication Manager Branch CTI objects (for example, CTI users, CTI links, and CTI sessions). To go to this screen, from the **Navigation** pane on the main menu click **Managed Objects > Maintenance & Monitoring > Telephony > CTI**.

Figure 13: Branch Device Manager CTI Maintenance screen



This screen provides:

- A **Restart Service** button to restart the CTI services.
- Status data for the switch and the CTI links. This data includes:
 - Whether Communication Manager Branch application enablement is properly running and connected to other system components on the Communication Manager Branch. If this is the case, both **Switch** and **CTI Link** are listed as talking.
 - The time that the links entered the current service state.
 - Message counts in the last half hour.
 - The number of associations (active CTI sessions).

Note:

The number of ASAI associations includes all active associations. This number should be stable, generally reflecting the number of station monitors, but will occasionally fluctuate since actively monitored calls and other transient CTI queries are included at the moment the status page is updated.

- A table that shows the CTI user connections. This data includes:
 - The CTI user name that was used to authenticate.
 - The IP address of the application computer.
 - The time that the application session began.
- Security data that shows if the client connection is using an encrypted or non-encrypted link.

For information about using this screen, see the **CTI** page in the Branch Device Manager Online Help.

Chapter 6: Troubleshooting CTI Problems

The following sections describe how to troubleshoot CTI problems. In the first section, you can find information on troubleshooting problems with TAPI-based CTI on Communication Manager Branch. Although similar problems can occur with TSAPI, for TSAPI-based applications follow procedures described in [Troubleshooting Problems with TSAPI-based CTI](#). Use care to follow each step in order without skipping steps to be sure the problem can be diagnosed properly.

Troubleshooting Problems with TAPI-based CTI

- [CTI TAPI-based applications are not functioning](#) on page 49
- [CTI TAPI-based applications respond slowly to TAPI requests](#) on page 58
- [CTI TAPI-based applications exhibit a specific telephony feature failure](#) on page 59

CTI TAPI-based applications are not functioning

Use the following steps to troubleshoot a CTI TAPI-based application that is not functioning:

1. Can you call out and receive calls from the telephone that you are trying to control?
 - Yes: Go to step [2](#).
 - No: This is not an application problem. For information on how to troubleshoot this problem, see *Maintenance and Troubleshooting for Avaya Aura™ Communication Manager Branch* (03-602039).
2. Does the application function on another PC?
 - Yes: Go to step [5](#).
 - No: Go to step [3](#).
3. Log on to Branch Device Manager and click **CTI** under **Managed Objects > Maintenance and Monitoring > Telephony**.

The CTI Maintenance screen appears as shown in example [Figure 14](#).

Figure 14: Computer Telephony Integration screen

CTI Maintenance

Computer Telephony Integration

Restart Service

Switch:	Talking	Messages To Switch:	16
Since:	2007-03-19 14:00:27 MDT	Messages From Switch:	16
CTI Link:	Talking	Number of Associations:	0
Since:	2007-03-19 14:00:27 MDT		

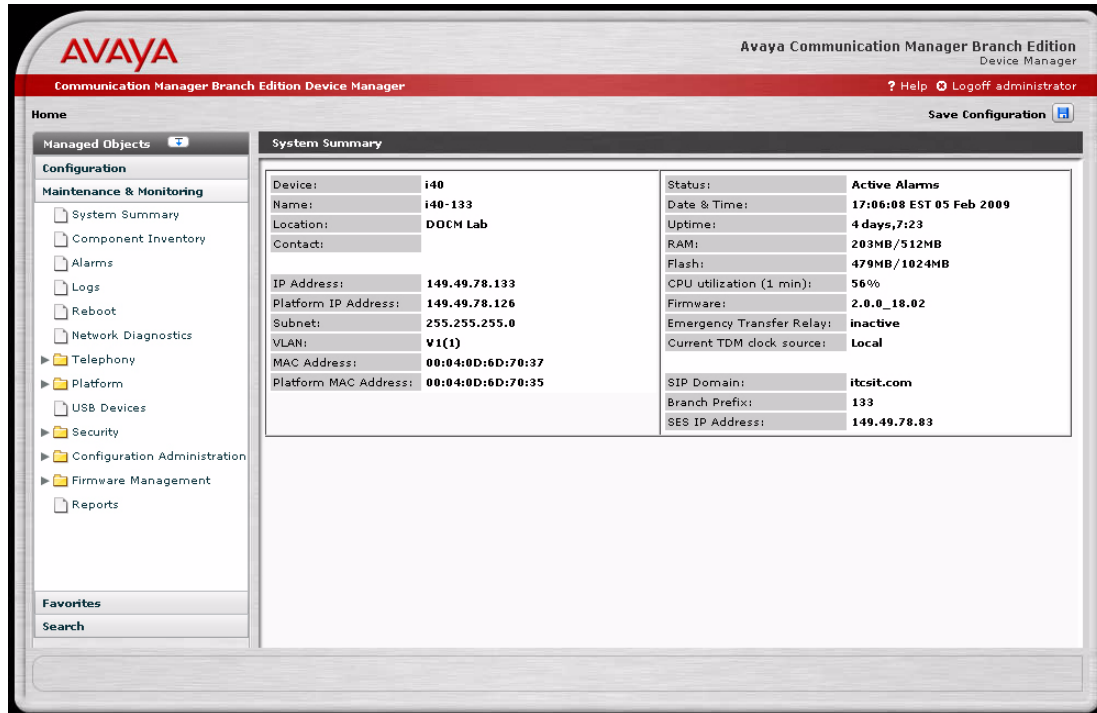
CTI Users Table

Username	IP Address	Time Opened
administrator	135.8.111.99	2007-03-09 12:03:50 EDT

Are the **Switch** and the **CTI Link** both in a **Talking** state?

- Yes: Continue to step [4](#).
 - No: Click **Restart Service**. Allow approximately one minute for the restart to complete and then refresh the screen. After the restart, verify that both the **Switch** and the **CTI link** are:
 - Talking: go to step [4](#).
 - Not talking: Contact your support organization.
4. From the Branch Device Manager interface, click **System Summary** under **Maintenance and Monitoring**. The **System Summary** screen appears as shown in example [Figure 15](#). Record the IP address of Communication Manager Branch in the **IP Address** field and the system name in the **Name** field.

Figure 15: System Summary screen



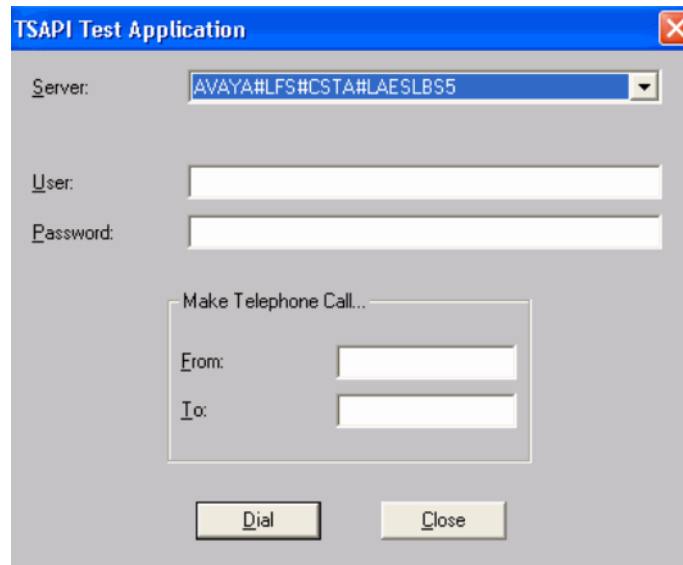
From a Windows command prompt on the application computer, execute the **ping** utility using the Communication Manager Branch IP address that you just recorded from the **System Summary** screen.

Was **ping** successful?

- Yes: Continue to step [5](#).
- No: Resolve the network issues between the application computer and Communication Manager Branch.

- From the Windows Start menu on the application computer, click **Start > All programs > Avaya AE Services > TSAPI Client > TSAPI Test**. The **TSAPI Test Application** screen appears as shown in example [Figure 16](#).

Figure 16: TSAPI Test Application screen



In the TSAPI Test Application screen, verify that **AVAYA#LFS#CSTA#systemname** (where **systemname** is the name of the Communication Manager Branch system) appears in the **Server** field or is available in the drop-down list.

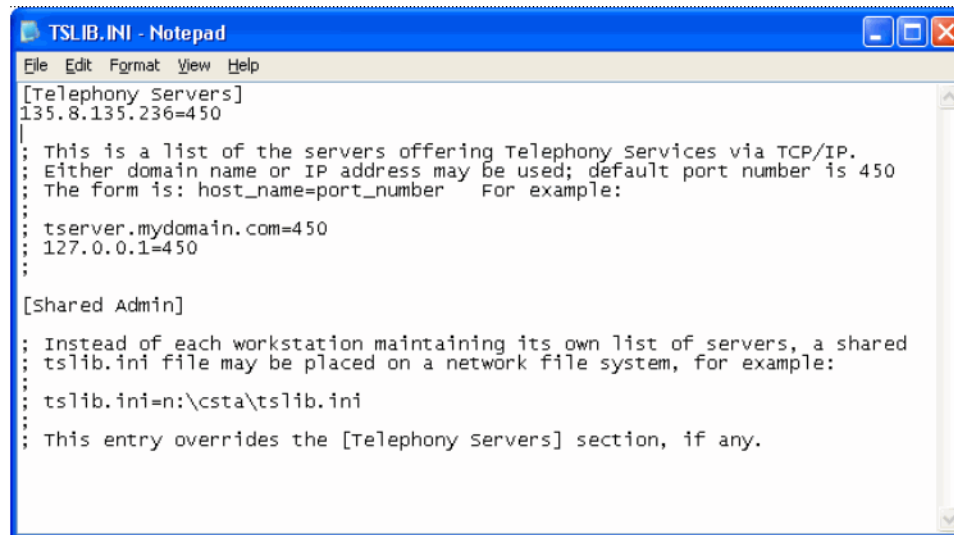
Note:

If you do not know the system name, click **System Summary** under **Maintenance and Monitoring** on the Branch Device Manager interface. The **System Summary** screen appears as shown in example [Figure 15](#). The system name displays in the **Name** field.

Does **AVAYA#LFS#CSTA#systemname** appear?

- Yes: Go to step [6](#).
- No: On the application computer, click **Start > all programs > Avaya AE Services > TSAPI Client > Edit TSLIB.INI**. The TSLIB.INI notepad appears as shown in [Figure 17](#).

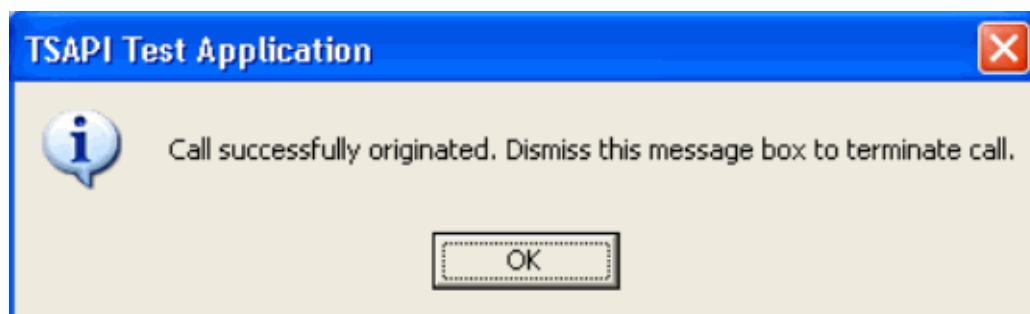
Figure 17: TSLIB.INI Notepad



Is there a line in the **Telephony Servers** section containing the recorded IP address of Communication Manager Branch with =450 on the end?

- Yes: Contact your support organization.
 - No: Then add a line containing the IP address and =450 to the **TSLIB.INI** file under the **Telephony Servers** heading similar to the example in [Figure 17](#). Save and close the **TSLIB.INI** file. Close the TSAPI Test Application. Repeat Step 5 from the beginning to verify that you successfully added the line to the **TSLIB.INI** file.
6. In the TSAPI Test Application, select the Communication Manager Branch server from the drop-down list and enter a valid CTI user and password. In the **Make a Telephone Call** section, enter your telephone extension in the **From** field and enter another working extension number in the **To** field. Click **dial**. If the call was successful the following window appears:

Figure 18: TSAPI Test Application window



Did the call complete and the **TSAPI Test Application** window appear?

- Yes: Close the **TSAPI Test Application window** by clicking **OK** and go to step [7](#).
- No: Find the error message that you received in [Table 3](#) and follow the suggested resolution.

Note:

If you do not find the error in [Table 3](#), contact your support organization.

Table 3: TSAPI Test Application error messages

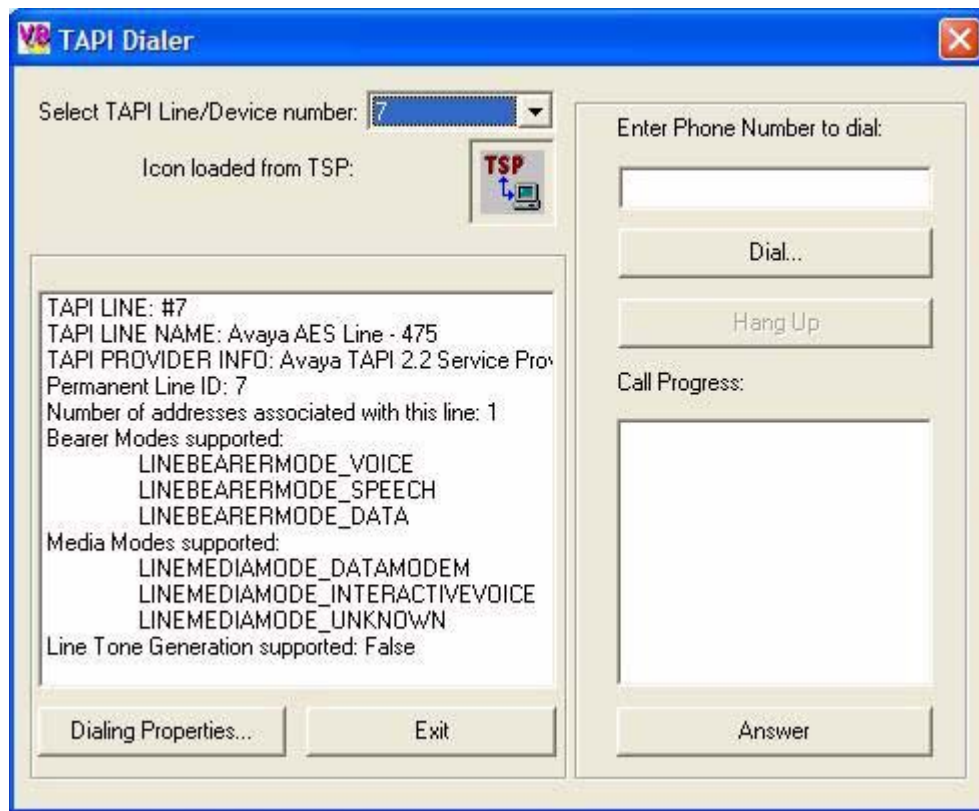
Error message	Meaning of Error Message	Resolution
Expecting Open Stream Conf, Received ACS Failure Conf: Bad Login or Password (25)	This error indicates that the login or user you have entered is not a valid CTI user.	Verify that you entered the proper login or user name when you executed the TSAPI test and that it was properly administered on Communication Manager Branch or the Active Directory Server.
Expecting Make Call Conf, received CSTA Failure Conf: invalid csta device identifier (12)	This error indicates that you have attempted to call from or to an invalid extension number.	Verify the extension numbers that you are using.
Expecting Make Call Conf, received CSTA Failure Conf: resource busy (33)	This error indicates that you have attempted to call from an extension that is already on an active call.	Clear the call from the extension and try again.
Expecting Make Call Conf, received CSTA Failure Conf: generic state incompatibility (21)	This error indicates that you have attempted to call from an extension that cannot go off hook to originate calls. It may be an analog phone or a phone administered without speakerphone capabilities.	Take the handset off the hook and wait for the dial tone to time-out and then try again.

7. From the application computer, click

Start > all programs > Avaya AE Services > TAPI Service Provider > TAPI Dialer.

The **TAPI Dialer** window appears as shown in example [Figure 19](#).

Figure 19: TAPI Dialer window



Try different selections for the **Select TAPI Line/Device number** drop-down list until you find the TAPI Line with the **TAPI Line Name** of **Avaya AES Line - XXX** (see [Figure 19](#) for an example). Did you find the line number with the **TAPI Line Name** of **Avaya AES Line**?

Note:

XXX is the extension number of the phone from which you expect to call. There can be several numbers shown.

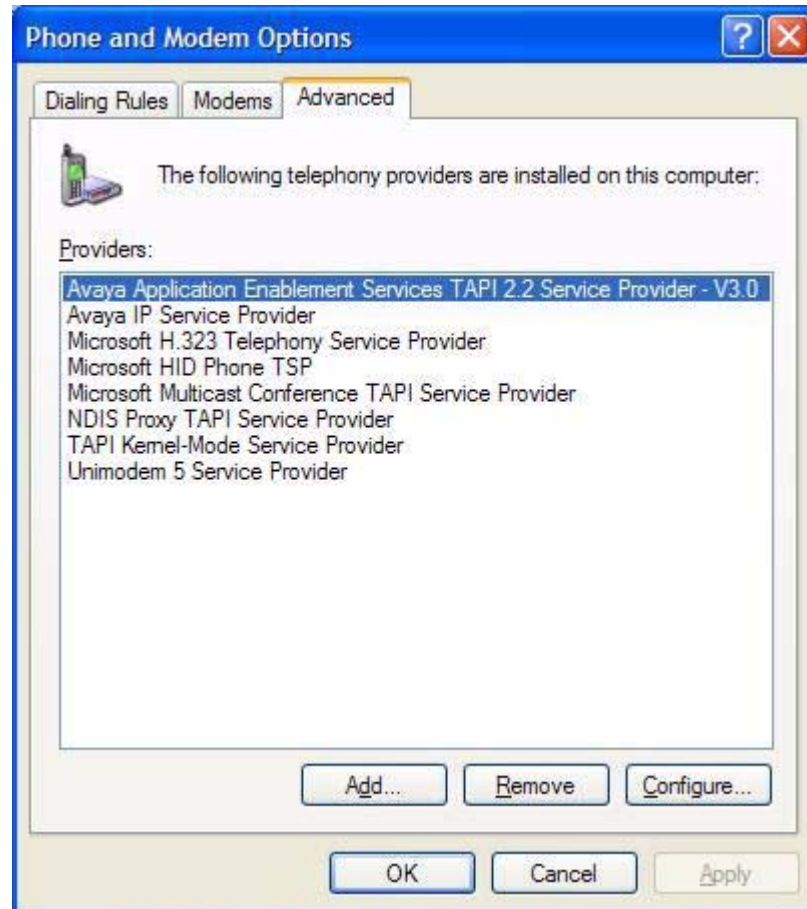
- Yes: Enter a known working local extension number other than your own in the **Enter Phone Number to Dial** box and click **Dial**.

Did the call complete and the **Hang Up** button become active?

- Yes: The Avaya TAPI TSP appears to be configured and functioning properly. If your application problem still exists, contact your third-party application vendor for assistance.
- No: Go to step [8](#).
- No: Go to step [8](#).

- From the Windows start menu, click **Settings > Control Panel > Phone and Modem Options**. Select the **Advanced** tab. The **Advanced** options tab displays as shown in example [Figure 20](#).

Figure 20: Phone and Modem Options window



Does the Avaya TSP appear on the list?

- Yes: Highlight the name and click **Configure**. The TSP Configuration window appears as shown in example [Figure 21](#). Follow the instructions that appear after [Figure 21](#).
- No: Click **Add**. If the Avaya TSP appears on the list, highlight it, click **Add** and proceed. If you are unable to add the provider, un-install and then re-install the Avaya TAPI TSP.

Can you add the provider?

- Yes: If the provider appears on the list, highlight the name and click **Configure**. The **TSP Configuration** window appears as shown in example [Figure 21](#). Follow the instructions that appear after [Figure 21](#).
- No: Call your support organization for help.

Figure 21: TSP Configuration window

The image shows the 'Avaya TSP Setup' window. It has a blue title bar with the text 'Avaya TSP Setup' and a close button. The window is divided into three main sections: 'Telephony connection', 'Valid devices', and 'Error logging'. In the 'Telephony connection' section, there is a 'Link Name' dropdown menu with the value 'AVAYA#LFS#CSTA#LAESLBS2', a 'Load' button, a 'User name' text box with 'johnsmith', a 'User password' text box with '*****', and a 'Test' button. The 'Valid devices' section has a 'Device list (separate multiple entries with a comma)' text box with the value '400,401,402,403-420'. The 'Error logging' section has an 'Error log extension' text box with 'AvayaTSP30.log', an 'Error log level' dropdown menu with 'Minor', an 'Error log path' text box with 'C:\Program Files\Avaya\AE Services\TAPI Service Provider\', and a 'Browse' button. At the bottom of the window are 'OK', 'Cancel', and 'About' buttons.

Verify that the:

- **Link Name** field contains **AVAYA#LFS#CSTA#systemname** or **AVAYA#LFS#CSTA-S#systemname** (where **systemname** is the name of the Communication Manager Branch system recorded earlier from the Branch Device Manager **System Summary** screen).
- **User name** and **User Password** fields contain valid information.
- Device list field contains one or more phone extensions that your TAPI-based application is attempting to monitor or control.

If the above settings are correct, reboot the application computer. After the reboot, repeat step 7 and attempt to make a call with the TAPI Dialer application.

Where you able to make a call using the TAPI Dialer application?

- Yes: The Avaya TAPI TSP appears to be configured and functioning properly. If your application problem still exists, contact your third party application vendor for assistance.
- No: Contact your support organization.

CTI TAPI-based applications respond slowly to TAPI requests

Use the following steps to troubleshoot a CTI TAPI-based application with slow response to TAPI requests:

1. Check the application computer hardware requirements against the minimum system requirements for the installed operating system, third party TAPI application, and the Avaya TAPI TSP.

Does the application computer meet the minimum hardware requirements?

- Yes: Go to step [2](#).
- No: Modify or replace the application computer to meet the minimum requirements.

2. Check the performance of the LAN between the application computer and Communication Manager Branch.

Is the network performing acceptably?

- Yes: Go to step [3](#).
- No: Troubleshoot and correct LAN problems.

3. Use Windows Performance Monitor to check for a excessive processor usage on the application computer.

Do you observe a sustained computer resource level of 70% or higher for more than five seconds?

- Yes: Go to step [4](#).
- No: Contact your support organization for assistance.

4. If you still have excessive processor usage, shut down the TAPI-based application. Using Windows Performance Monitor, check again for excessive processor usage on the application computer.

Do you still observe excessive computer resources being used without the TAPI-based application running?

- Yes: The excessive usage appears to be related to something other than the application. Troubleshoot and eliminate the reason for the excessive processor usage.
- No: Contact your third-party application vendor for assistance.

CTI TAPI-based applications exhibit a specific telephony feature failure

Use this section to troubleshoot a CTI TAPI-based application that is working but has a specific feature failure such as can't transfer, can't answer, etc.

1. Does the problem occur when you use the phone to manually perform the same function as the application?
 - Yes: The issue with the feature is not an application enablement problem. For information on Communication Manager Branch troubleshooting, see *Maintenance and Troubleshooting for Avaya Aura™ Communication Manager Branch*(03-602039).
 - No: Go to step [2](#).
2. Does the TAPI-based application exhibit the same problem on other application computers in the office?
 - Yes: Go to step [3](#).
 - No: Contact your third-party application vendor for assistance.
3. Consult the list of tested TAPI-based applications and their known functional outages in [Appendix A: Tested Applications](#) on page 87.

Is the problem on the list?

 - Yes: The application functionality is not supported at this time.
 - No: Go to step [4](#).
4. Contact your third-party application vendor for assistance.

Troubleshooting Problems with TSAPI-based CTI

This section can be used to troubleshoot problems with TSAPI-based CTI on Communication Manager Branch.

Use care to follow each step in order without skipping steps to be sure the problem can be diagnosed properly.

[TSAPI-based CTI applications are not functioning](#)

[TSAPI-based CTI applications respond slowly to TSAPI requests](#)

[TSAPI-based CTI applications exhibit a specific telephony feature failure](#)

TSAPI-based CTI applications are not functioning

Use the following steps to troubleshoot a TSAPI-based CTI application that is not functioning:

1. Can you call out and receive calls from the telephone that you are trying to control?
 - Yes: Go to step 2.
 - No: This is not an application problem. For information on how to troubleshoot this problem, see [Maintenance and Troubleshooting for Avaya Aura™ Communication Manager Branch\(03-602039\)](#).
2. Does the application function on another PC?
 - Yes: Go to step 5.
 - No: Go to step 3.
3. Log on to Branch Device Manager and click **CTI** under **Managed Objects > Maintenance & Monitoring > Telephony**.

The Computer Telephony Integration screen appears as shown in [Figure 22](#).

Figure 22: Computer Telephony Integration screen

The screenshot shows the 'CTI Maintenance' window. Under 'Computer Telephony Integration', there is a 'Restart Service' button. Below this, there are two status sections. The first section shows 'Switch:' as 'Talking' since '2007-03-19 14:00:27 MDT' and 'CTI Link:' as 'Talking' since '2007-03-19 14:00:27 MDT'. The second section shows 'Messages To Switch:' as '16', 'Messages From Switch:' as '16', and 'Number of Associations:' as '0'. At the bottom, there is a 'CTI Users Table' with one user: 'administrator' with IP address '135.8.111.99' and 'Time Opened' '2007-03-09 12:03:50 EDT'.

CTI Users Table		
Username	IP Address	Time Opened
administrator	135.8.111.99	2007-03-09 12:03:50 EDT

Are the **Switch** and the **CTI Link** both in a **Talking** state?

- Yes: Continue to step 4.
- No: Click **Restart Service**. Allow approximately one minute for the restart to complete and then refresh the screen. After the restart, verify that both the **Switch** and the **CTI link** are:
 - Talking: go to step 4.

- Not talking: Contact your support organization.
4. From the Branch Device Manager interface, click **System Summary** under **Managed Objects > Maintenance and Monitoring**. The **System Summary** screen appears as shown in [Figure 23](#). Record the IP address of Communication Manager Branch in the **IP Address** field and the system name in the **Name** field.

Figure 23: System Summary screen

System Summary			
Device:	i40	Status:	Active Alarms
Name:	i40-133	Date & Time:	17:06:08 EST 05 Feb 2009
Location:	DOCM Lab	Uptime:	4 days,7:23
Contact:		RAM:	203MB/512MB
IP Address:	149.49.78.133	Flash:	479MB/1024MB
Platform IP Address:	149.49.78.126	CPU utilization (1 min):	56%
Subnet:	255.255.255.0	Firmware:	2.0.0_18.02
VLAN:	V1(1)	Emergency Transfer Relay:	inactive
MAC Address:	00:04:0D:6D:70:37	Current TDM clock source:	Local
Platform MAC Address:	00:04:0D:6D:70:35	SIP Domain:	itscit.com
		Branch Prefix:	133
		SES IP Address:	149.49.78.83

From a Windows command prompt on the application computer, execute the **ping** utility using the Communication Manager Branch IP address that you just recorded from the **System Summary** screen.

Was ping successful?

- Yes: Continue to step 5.
 - No: Resolve the network issues between the application computer and Communication Manager Branch.
5. From the Windows Start menu on the application computer, click **Start > All programs > Avaya AE Services > TSAPI Client > TSAPI Test**. The **TSAPI Test Application** screen appears as shown in .

Figure 24: TSAPI Test Application screen

In the TSAPI Test Application screen, verify that **AVAYA#LFS#CSTA#systemname** or **AVAYA#LFS#CSTA-S#systemname** (where systemname is the name of the Communication Manager Branch system) appears in the Server field or is available in the drop-down list.

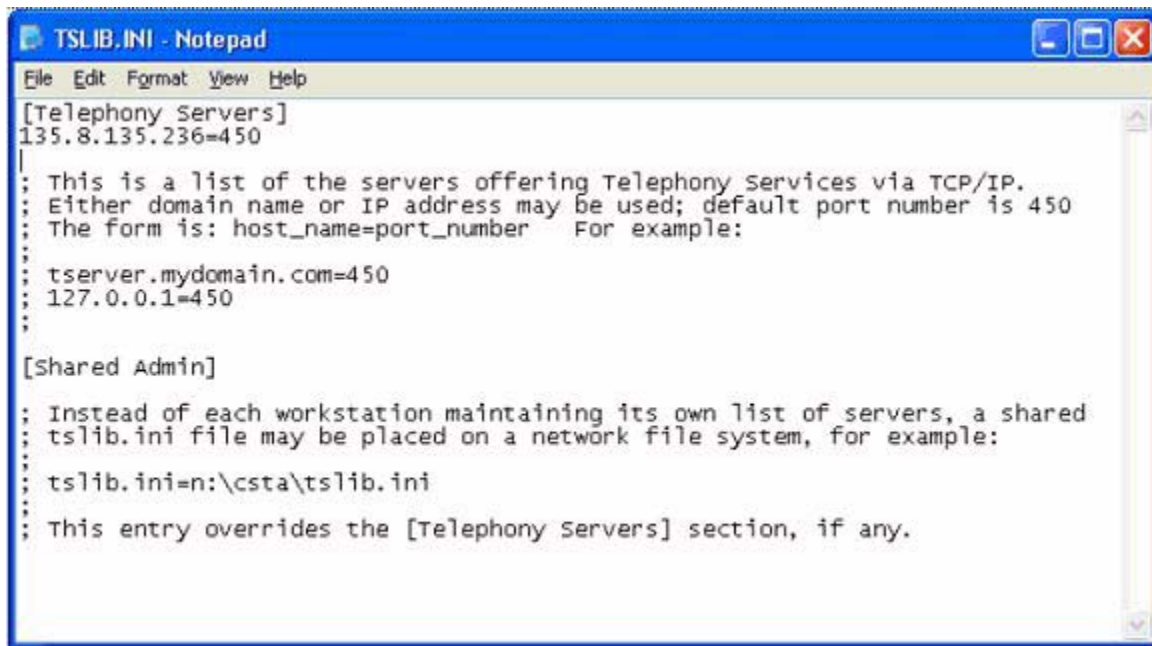
Note:

If you do not know the system name, click **System Summary** under **Managed Objects > Maintenance and Monitoring** on the Branch Device Manager interface. The **System Summary** screen appears as shown in [Figure 23](#). The system name displays in the **Name** field.

Does **AVAYA#LFS#CSTA#systemname** appear?

- Yes: Go to step 6.
- No: On the application computer, click **Start > All programs > Avaya AE Services > TSAPI Client > Edit TSLIB.INI**. The TSLIB.INI notepad appears as shown in [Figure 25](#).

Figure 25: TSLIB.INI Notepad



Is there a line in the **Telephony Servers** section containing the recorded IP address of Communication Manager Branch with =450 on the end?

- Yes: Contact your support organization.
 - No: Then add a line containing the IP address and =450 to the **TSLIB.INI** file under the **Telephony Servers** heading similar to the example in Figure 11. Save and close the **TSLIB.INI** file. Close the TSAPI Test Application. Repeat Step 5 from the beginning to verify that you successfully added the line to the **TSLIB.INI** file.
6. In the TSAPI Test Application, select the Communication Manager Branch server from the drop-down list and enter a valid CTI user and password. In the **Make a Telephone Call** section, enter your telephone extension in the **From** field and enter another working extension number in the **To** field. Click **dial**. If the call was successful [Figure 26](#) appears:

Figure 26: TSAPI Test Application window (Success)



Did the call complete and the TSAPI Test Application window appear?

- Yes: Close the TSAPI Test Application window by clicking OK and go to step 7.
- No: Find the error message that you received in Table 3 and follow the suggested resolution.

Note:

If you do not find the error in [Table 4](#), contact your support organization.

Table 4: TSAPI Test Application Error Messages

Error message	Meaning of Error Message	Resolution
Expecting Open Stream Conf, Received ACS Failure Conf: Bad Login or Password (25)	This error indicates that the login or user you have entered is not a valid CTI user.	Verify that you entered the proper login or user name when you executed the TSAPI test and that it was properly administered on Distributed Office or the Active Directory Server.
Expecting Make Call Conf, received CSTA Failure Conf: invalid csta device identifier (12)	This error indicates that you have attempted to call from or to an invalid extension number.	Verify the extension numbers that you are using.
Expecting Make Call Conf, received CSTA Failure Conf: resource busy (33)	This error indicates that you have attempted to call from an extension that is already on an active call.	Clear the call from the extension and try again.
Expecting Make Call Conf, received CSTA Failure Conf: generic state incompatibility (21)	This error indicates that you have attempted to call from an extension that cannot go off hook to originate calls. It may be an analog phone or a phone administered without speakerphone capabilities.	Take the handset off the hook and wait for the dial tone to time-out and then try again.

7. Were you able to place a call successfully with the **TSAPI Test Application**?

- Yes: The Avaya TSAPI Client appears to be configured and functioning properly. If your application problem still exists, contact your third-party application vendor for assistance.
- No: Contact your support organization.

TSAPI-based CTI applications respond slowly to TSAPI requests

Use the following steps to troubleshoot a TSAPI-based CTI application with slow response to TSAPI requests:

1. Check the application computer hardware requirements against the minimum system requirements for the installed operating system, third party TSAPI application, and the Avaya TSP.

Does the application computer meet the minimum hardware requirements?

- Yes: Go to step 2.
- No: Modify or replace the application computer to meet the minimum requirements.

2. Check the performance of the LAN between the application computer and Communication Manager Branch.

Is the network performing acceptably?

- Yes: Go to step 3.
- No: Troubleshoot and correct LAN problems.

3. Use Windows Performance Monitor to check for a excessive processor usage on the application computer.

Do you observe a sustained computer resource level of 70% or higher for more than five seconds?

- Yes: Go to step 4.
- No: Contact your support organization for assistance.

4. If you still have excessive processor usage, shut down the TSAPI-based application. Using Windows Performance Monitor, check again for excessive processor usage on the application computer.

Do you still observe excessive computer resources being used without the TSAPI-based application running?

- Yes: The excessive usage appears to be related to something other than the application. Troubleshoot and eliminate the reason for the excessive processor usage.
- No: Contact your third-party application vendor for assistance.

TSAPI-based CTI applications exhibit a specific telephony feature failure

Use this section to troubleshoot a TSAPI-based CTI application that is working but has a specific feature failure such as can't transfer, can't answer, etc.

Troubleshooting CTI Problems

1. Does the problem occur when you use the phone to manually perform the same function as the application?
 - Yes: The issue with the feature is not an application enablement problem. For information on Communication Manager Branch troubleshooting, see [*Maintenance and Troubleshooting for Avaya™ Communication Manager Branch\(03-602029\)*](#).
 - No: Contact your third-party application vendor for assistance, if applicable, and/or your support organization

Chapter 7: Programmer's Reference Information

This chapter contains detailed technical information that will be of interest primarily to software developers or those interested in evaluating whether an existing application can be ported to a Communication Manager BranchR2.0 environment.

General Characteristics of Communication Manager Branch CTI Support

Although Communication Manager Branch supports both TAPI and TSAPI, the specific capabilities of Communication Manager Branch greatly determine which services and events specified in these APIs can actually be used, and when used, what detailed behaviors arise. For example, even though TSAPI itself supports service requests to initiate call-monitoring-via-device, since this service requires call vectors that are not provided in Communication Manager BranchR2.0, the requested function cannot be used in a Communication Manager Branch environment. Such a limitation caused by the Communication Manager Branch server itself will apply whether the request is made through TAPI or TSAPI. In this section, general characteristics and limitations of Communication Manager Branch that affect both TAPI and TSAPI are presented.

Multi-application interaction

Client application computers can run multiple CTI-enabled applications, up to the capacity of the client. Further, a single client application computer can run both TAPI and TSAPI applications simultaneously.

Multiple applications can monitor the same endpoint. However, there will be restrictions on simultaneous control of the same endpoint by multiple applications. Competing applications cannot successfully request the same endpoint to make simultaneous calls, for example.

Multiple applications can monitor the same call simultaneously. However, only one application at a time can have control of the call. An application must control a call for an action to be taken on the call. Actions include transferring it, adding or removing parties to/from it, putting it on hold at a particular device, etc.

CTI-controllable endpoints

SIP devices

Do not monitor SIP devices for CTI events. SIP messages at these devices are not translated consistently into TAPI or TSAPI messages. These devices cannot be monitored, snapshots taken, or controlled by CTI services. However, SIP devices can be specified in CTI service requests as "target" or "destination" devices. For example, a call can be placed or transferred via CTI to such a device. In addition, you can add a SIP device to an existing call via TAPI and TSAPI CTI Conference services.

Note:

Some devices (16xx-series phones) can be operated either as SIP or H.323 endpoints. When in H.323 mode CTI can monitor and control these devices.

Key configuration endpoints

"Key configuration" endpoints are physical phones or CTI Extensions that have Line Appearances assigned to buttons. Key configuration endpoints may or may share the LAs.

Communication Manager Branch supports many features typical of telephone key systems using key configuration endpoints. In Communication Manager Branch2.0 key configuration endpoints can be controlled and monitored via TAPI and TSAPI CTI services and events.

However, not all key system operations that can be accomplished manually or through feature access codes can be accomplished via CTI services. For example, a specific LA cannot be selected to be answered or used for an outgoing call. When a CTI-based request to answer or make a call is made, the Communication Manager Branch will select an alerting call to be answered or an outgoing LA for an outgoing call. Additional CTI capabilities limitations applied to key configuration endpoints will be noted in the TAPI and TSAPI sections that follow.

The following table shows the TSAPI and TAPI services that can be applied to Line Appearances (as well as Call Appearances):

Table 5: TSAPI and TAP services

TSAPI	TAPI	Comments
Alternate Call	lineSwapHold	On-hold and Connected call must both be present and known
Answer Call	lineAnswer	
Clear Call		
1 of 2		

Table 5: TSAPI and TAP services

TSAPI	TAPI	Comments
Clear Connection	lineDrop	
Conference Call • Two-step • Single-step	lineSetupConference lineAddToConference	
Deflect	lineRedirect	
Hold Call	lineHold	
Pickup Call		
Reconnect Call		
Retrieve Call	lineUnhold	
Transfer Call • Two-step • Single-step	lineBlindTransfer lineSetupTransfer lineCompleteTransfer	
		2 of 2

CTI Extensions

Communication Manager Branch R2.0 support CTI Extensions. A CTI Extension is a special logical-only station type (there is no physical hardware) that can be created in Communication Manager Branch software. A CTI Extension is created by assigning "CTI" as a station type in User Administration when an extension is assigned to a user. A CTI Extension acts like a physical phone in that CTI services (such as originating a call) can be applied to it and it can accept or terminate incoming calls like a physical phone. A CTI Extension can have both Call and Line appearances like a physical phone. Because they do not have physical hardware, CTI Extensions can be created to conveniently test existing or new CTI-enabled applications without the cost or time involved in installing and setting up physical hardware phones. CTI Extensions can also be used creatively by CTI-enabled application developers to implement customized or novel system features.

CTI-affecting Communication Manager Branch characteristics

Certain CTI capabilities that are exposed through TAPI and TSAPI are not available on Communication Manager Branch servers because the server does not offer the required underlying services or events. [Table 6](#) lists certain features that are available on other Avaya

platforms but not on Communication Manager Branch which limit or otherwise affect CTI capabilities actually available through TAPI or TSAPI.

Table 6: TSAPI and TAP services

Avaya Platform Capability not Available on Communication Manager Branch	TSAPI Impact	TAPI Impact
Vectors	Monitor-Call-Via-Device not provided	
	Queued events not supported	Queued events not supported
		No TAPI 2.2 Call Center capabilities are supported
DTMF Digit collection via vector steps not supported	Get Digits/Digits Collected not supported	No DTMF digit collection capabilities

CTI-specific alarms on the Communication Manager Branch server

The Branch Device Manager terminal's alarm display presents alarms arising from Communication Manager Branch's CTI functions. CTI-related alarms are automatically cleared when the reported alarm condition clears.

Reliability and recovery

Communication Manager Branch supports a client-to-server heartbeat function. Client computers regularly transmit heartbeat messages regularly to the Communication Manager Branch server. These messages will reveal if a CTI link or the Communication Manager Branch server CTI functions becomes non-operational without any need to actually request a service.

Using TAPI

Communication Manager Branch2.0 provides a subset of TAPI 2.2 that is compliant with the Microsoft specification. Visit the Microsoft TAPI developer's website.

[http://msdn2.microsoft.com/en-us/library/ms737220\(VS.85\).aspx?wt.svl=overview](http://msdn2.microsoft.com/en-us/library/ms737220(VS.85).aspx?wt.svl=overview)

for detailed information on TAPI services and events and their constituent parameters as well as basic information on how to use TAPI. The website provides information on how events are received, how to access the information they contain in data structures, how parameters in service requests are populated, and so forth. Examples of TAPI code are also provided.

This section provides information on the specific subset of TAPI services, events, call states and data structures that are supported by the Communication Manager Branch2.0 TAPI implementation.

Supported TAPI capability classes

TAPI application support includes each of the following Microsoft-defined capability classes, although not all capabilities within each class are available. For a list of supported TAPI services and events, see Appendix B: Supported TAPI Services and Events.

- "Assisted Telephony. A small set of TAPI services to support simple CTI interaction between two applications.
- "Basic Telephony. A set of basic telephony services that includes Make Call, Answer Call, and Drop Call.
- "Supplementary Telephony Services. A set of TAPI functions that includes Transfer, Hold, Retrieve, and Conference.
- "TAPI Extended Services are **not** supported

Supported TAPI service requests

[Table 7](#) lists the TAPI service methods that are supported by Communication Manager BranchR2.0. Distributed Office R1.1 also supports this list, with one additional method, lineRedirect, having been added.

Table 7: Supported TAPI Service

TAPI function	Comments
tapiGetLocationInfo	
tapiRequestDrop	
tapiRequestMakeCall	
lineAddToConference	
1 of 3	

Table 7: Supported TAPI Service

lineAnswer	
lineBlindTransfer	
lineClose	
lineCompleteTransfer	
lineDeallocateCall	
lineDial	
lineDrop	
lineGetAddressCaps	
lineGetAddressStatus	
lineGetCallInfo	
lineGetCallStatus	
lineGetDevCaps	
lineGetIcon	
lineGetID	
lineGetLineDevStatus	
lineHold	
lineInitializeEx	
lineMakeCall	
lineNegotiateAPIVersion	
lineOpen	
lineRedirect	
lineSetStatusMessages	
lineSetupConference	
lineSetupTransfer	
lineShutdown	
lineSwapHold	
2 of 3	

Table 7: Supported TAPI Service

lineTranslateAddress	
lineUnhold	
3 of 3	

Supported TAPI events

[Table 8](#) lists the TAPI events that Communication Manager BranchR2.0 TAPI TSP support. Note that in some cases the underlying system event or status (such as a call being placed on hold or being on hold) is indicated by providing a specific constant (e.g. LINECALLSTATE_ONHOLD) in a generic event message (e.g., LINE_CALLSTATE). The TAPI TSP supports the ability to inform the application when these system events occur even though there is not a separate "Event Message" for each.

Table 8: Supported TAPI Events

Event Message	Supported Event Constants	Description
LINE_CALLSTATE		Notifies the application that the status of the specified call has changed. Typically, several such messages are received during the lifetime of a call. Also informs the application of new incoming calls in the offering state. The application can use <i>lineGetCallStatus</i> to retrieve more information about the current status of the call.
	LINECALLSTATE_OFFERING	Notifies the application that the call is being offered to the station.
	LINECALLSTATE_CONNECTED	Notifies the application that the call has been established (connected). Information is now able to flow between the originating and destination addresses.
	LINECALLSTATE_DIALING	Notifies the application that the originator is dialing digits on the call. Digits are collected by the switch.
1 of 4		

Table 8: Supported TAPI Events

	LINECALLSTATE_DIALTONE	Notifies the application that the call is receiving a dial tone from the switch, which means the switch is ready to receive a dialed number. For details, refer to <i>LINE_DIALTONE_MODE</i> Constants in Microsoft Telephony Application Programming Interface (TAPI) Version 2.2.
	LINECALLSTATE_DISCONNECTED	Notifies the application that the remote party has disconnected from the call. For details, refer to <i>LINE_DISCONNECT_MODE</i> Constants in Microsoft Telephony Application Programming Interface (TAPI) Version 2.2.
	LINECALLSTATE_IDLE	Notifies the application that the call exists but is not connected (active). A call can never transition out of the idle state.
	LINECALLSTATE_PROCEEDING	Notifies the application that dialing is complete and the call is proceeding through the switch. Occurs before the call reaches the dialed party.
	LINECALLSTATE_RINGBACK	Notifies the application that the destination address is reached and the destination switch is returning a ringing tone to the originator. The destination address is alerted to the call.
	LINECALLSTATE_ONHOLD	Notifies the application that the call is on hold.
	LINECALLSTATE_BUSY	Notifies the application that the call is receiving a busy tone. A busy tone indicates a circuit (trunk) or the remote party's station is in use.
2 of 4		

Table 8: Supported TAPI Events

	LINECALLSTATE_CONFERENCED	Notifies the application that the call is a member of a conference call and is in the connected state. <i>LINE_CLOSE</i> Notifies the application that the specified line device has been forcibly closed. The line device handle or any call handles for calls on the line are no longer valid.
LINE_CREATE		Notifies the application of the creation of a new line device.
LINE_LINEDEVSTATE		Notifies the application that the state of a line device has changed. The application can invoke <i>lineGetLineDevStatus</i> to learn the line's new status
	LINEDEVSTATE_CLOSE	Notifies the application that the line has been closed by another application.
	LINEDEVSTATE_CONNECTED	Notifies the application that the line was previously disconnected and is now connected to TAPI
	LINEDEVSTATE_DISCONNECTED	Notifies the application that the line was previously connected and is now disconnected from TAPI.
	LINEDEVSTATE_REINIT	Notifies the application that items have changed in the configuration of line devices. To register the changes, the application should reinitialize its use of TAPI.

3 of 4

Table 8: Supported TAPI Events

	LINEDEVSTATE_REMOVE D	Notifies the application that a device is being removed from the system by the service provider (most likely through user action via a Control Panel or similar utility). The <i>LINE_CLOSE</i> event normally follows. If there is an attempt to access the device before the TAPI is reinitialized, the <i>LINEERR_NODEVICE</i> message returns to the application. If a service provider sends <i>LINE_LINEDEVSTATE</i> containing this value to TAPI, TAPI notifies any application negotiating TAPI version 1.4 or later.
	LINEDEVSTATE_RINGING	Notifies the application that the switch is instructing the line to alert the user.
LINE_REMOVE		Notifies the application that a line device has been removed (deleted) from the system. Generally, this is not used for temporary removals, such as extraction of PCMCIA devices, but for permanent removals in which the device would no longer be reported by the service provider if TAPI were reinitialized.
LINE_CALLINFO		The TAPI <i>LINE_CALLINFO</i> message is sent when the call information about the specified call has changed. The application can invoke <i>lineGetCallInfo</i> to determine the current call information. <i>CallerID</i> (originating party of an incoming call) and <i>CalledID</i> are available immediately after the call is offered to the application. Applications should not necessarily wait for <i>LINE_CALLINFO</i> messages before trying to retrieve this information.
LINE_REPLY		Notifies the application about the results of function calls that were completed asynchronously.

4 of 4

Supported TAPI data structures

The following TAPI structures are populated and accessible (via "get data" TAPI functions) to applications. This is a subset of all TAPI data structures.

Table 9: TAPI Data Structures

TAPI Data Structure

LINEADDRESSCAPS
LINEADDRESSSTATUS
LINECALLINFO
LINECALLPARAMS
LINECALLSTATUS
LINEDEVCAPS
LINEDEVSTATUS

Using TSAPI

Information in this section can be used on its own, but it is intended to compliment detailed information provided in the Avaya TSAPI Programmer's Reference Guide (TSAPI for Communications Manager Programmer Reference, Release 4.2.0). That PRG contains complete details of all ACS and TSAPI services and events in Avaya's TSAPI implementation. Although Communication Manager BranchR2.0 provides a complete implementation of Avaya TSAPI 4.2, you can not achieve all services and events specified on the Communication Manager BranchR2.0 server because the necessary server functions are not implemented. In the following sections, the ACS and TSAPI services and events that are functionally available through TSAPI on Communication Manager Branch2.0 are listed. Unless noted, all parameters specified in the TSAPI PRG are functionally supported.

Introduction

At a high-level, the processes by which a TSAPI CTI-enabled application accesses and uses communication services provided by DOae comprise the following steps:

1. The application first uses Application Control Services (ACS) to request and establish an active CTI session with the DOae server. The TSAPI CTI application provides a CTI User ID

and password when making this request. The DOae server will examine the provided credentials by checking them against CTI User IDs/passwords administered on the Communication Manager Branch server. If a CTI User ID/password match is made, the CTI session will be authorized and established. The DOae will provide a regular "heartbeat" ACS event to the application as long as the CTI session is active and functioning correctly.

2. Once the CTI session is established, the application can use TSAPI CSTA-based service requests to establish monitors on devices that will be observed and controlled through CTI events and services. Typically, once monitors are correctly enabled, the CTI-enabled application begins using various TSAPI CTI services (such as CSTA Make Call or CSTA Answer Call) to request the Communication Manager Branch server to perform various communication functions.
3. When the CTI-enabled application ends, it should use an ACS service to close its active CTI session. However, this is not mandatory, as DOae will eventually close inactive or failed CTI sessions that might remain should, for example, an application unexpectedly terminate.

Non-supported TSAPI Services and Events

Although the Communication Manager Branch TSAPI implementation is complete in that:

- An application may issue any TSAPI service request
- All parameters may be provided
- All parameters are included in events that may arrive

this does not ensure that all such requests will be fulfilled by Communication Manager Branch or that any specific TSAPI event will be generated by Communication Manager Branch. Whether a service request will be fulfilled or an event will be generated depends on the capabilities of the switching system - in this case Communication Manager Branch- to which TSAPI requests are issued and from which events are generated. Because of Communication Manager Branch's designed capabilities, only a limited set of TSAPI requests will be fulfilled and some TSAPI events will never be generated. In the case of service requests that cannot be fulfilled, an application will often receive a negative acknowledgement with an appropriate error code. In some cases, a service request cannot even be issued because the target object - for instance, a vector - cannot be created in Communication Manager Branch (via Communication Manager Branch administration). In the case of unsupported events, these events simply will never arrive and so an application will never need to deal with them.

Note:

An application that is being ported from another switching platform - such as Communication Manager - may need to deal with the absence of a needed event that is not supported on Communication Manager Branch.

The following TSAPI services and events are not supported by the Communication Manager Branch server:

Table 10: Unsupported TSAPI services and events

Unsupported TSAPI Service/Event	Comment
Consultation – Direct Agent Call Service	
Consultation – Supervisor Assist Call Service	
Make Direct Agent Call Service	
Make Predictive Call Service	
Make Supervisor Assist Call Service	
Selective Listening Hold Service	
Selective Listening Retrieve Service	
Set Advice of Charge Service	
Set Agent State Service	
Set Billing Rate Service	
Query ACD Split Service	
Query Agent Login Service	
Query Call Classifier Service	
Query Device Name Service	
Query Trunk Group Service	
Monitor Calls Via Device Service	Requires Vectors
Entered Digits Event	Requires Vectors
Logged Off Event	
Logged On Event	
Events arising from Attendant operations	

1 of 2

Table 10: Unsupported TSAPI services and events

Events arising from ACDs including Delivered to ACD device. Includes events arising from: • Auto-Available Splits • Night-service • Call coverage events from call monitors on ACDs	
Queued Event	Only generated from VDN
Route End Event	
Route End Service	
Route Register Abort Event	
Route Register Cancel Service	
Route Register Service	
Route Request Service	
Route Select Service	Neither TSAPI version 1 nor version 2
Route Used Event	Neither TSAPI version 1 nor version 2
System Status Request Event	
System Status Request Confirmation	
System Status Event Send service	
2 of 2	

Initiating and managing a TSAPI CTI session - ACS services and events

TSAPI applications initiate and manage CTI connection links to the Communication Manager Branch server in the standard way specified in the TSAPI Programmer's Reference - via ACS services and events. Once a stream is opened via the `acsOpenStream()` service, a stream reference is provided and TSAPI service requests and event messages may then be

exchanged with the Communication Manager Branch server. The following ACS services and events are supported:

Table 11: Support ACS Services and Events

TSAPI Service/Event	Comment
ACS	
acsOpenStream()	
ACSOpenStreamConfEvent	
acsCloseStream()	
ACSCloseStreamConfEvent	
acsAbortStream()	
acsGetEventBlock()	
acsGetEventPoll()	
acsSetESR()	Win32
acsEventNotify()	Win32
acsFlushEventQueue()	
acsEnumServerNames()	
ACSUniversalFailureEvent	

Control and environment discovery services and events

Table 12: Control and Environment Discovery Services and Events

TSAPI Service/Event	Comment
cstaGetAPICaps()	
CSTAGetAPICapsConfEvent	
cstaQueryCallMonitor()	
CSTAQueryCallMonitorConfEvent	

Call control services and events

Table 13: Call Control Services and Events

TSAPI Service/Event	Comment
Call Control Services Certain restrictions apply in the case of key-configured endpoints	
Alternate Call	
Answer Call	
Clear Call	
Clear Connection	
Conference Call	
Consultation Call	
Deflect Call	
Hold Call	
Make Call	
Pickup Call	
Reconnect Call	
Retrieve Call	
Send DTMF Tone	

1 of 2

Table 13: Call Control Services and Events

Single-step Conference	
Single-step Transfer	
Transfer Call	
Events (event details per 4.2.0 Programmer Reference, Chapter 11, 661-677 will be observed)	
Call Cleared	
Conferenced	
Connection Cleared	
Delivered	
Diverted	
Entered Digits	Private
Established	
Failed	
Held	
Network Reached	
Originated	
Retrieved	
Service Initiated	
Transferred	

2 of 2

Logical feature services

Table 14: Logical Feature Services

TSAPI Service	Comment
Set Do Not Disturb	
Set Forwarding	All forwarding conditions may be specified.
Set Message Waiting Indicator	

Query services

Table 15: Query Service

TSAPI Service/Event	Comment
Query Device Info	
Query Device Name	
Query Do Not Disturb	
Query Forwarding	
Query Message Waiting	
Query Station Status	
Query Time of Day	
Query Trunk Group	

Snapshot services

Table 16: Snapshot Services

TSAPI Service/Event	Comment
Snapshot Call	States of calls at LAs and CAs reported properly
Snapshot Device	States of calls at LAs and CAs reported properly

Monitor services and events

Table 17: Monitor Services and Events

TSAPI Service/Event	Comment
Change Monitor Filter	
Monitor Call	Local Connection parameter supported for LA and CA calls Monitor Call Via Device not supported
Monitor Device	localConnection parameter supported for LA and CA calls
Monitor Ended Event	
Monitor Stop on Call	Private
Monitor Stop	

System status services and events

Table 18: System Status Services and Events

TSAPI Service/Event	Comment
System Status Request	
System Status Start	
System Status Stop	
Change System Status Filter	
System Status Event	

Universal failure events

Table 19: Universal Failure Events

Event	Comment
ACSUniversalFailureConfEvent	

Appendix A: Tested Applications

Avaya Aura Communication Manager Branch application enablement has been tested and works with the applications listed in [Table 20: Tested applications](#). This is not a complete list of supported applications, other applications may work with Communication Manager Branch application enablement depending on which individual TAPI services and events are used. See [Appendix B: Supported TAPI Services and Events](#) for guidance. Further information may be obtained from the DevConnect team.

Table 20: Tested applications

Application	Versions	Observed Limitations
Act!	2006	Features not supported: Transfer, Conference. Application does not detect existing calls on startup.
Act! with Act!Tapi.Net		
GoldMine	6.7	
Goldmine with Tapi Link		
Maximizer	9	
Microsoft Outlook	2000 or later	
Microsoft Phone Dialer	1.5	

Tested Applications

Appendix B: Supported TAPI Services and Events

TAPI-based applications can invoke each of the TAPI 2.1 and 2.2 services shown in [Table 21: Supported TAPI services](#).

Table 21: Supported TAPI services

Service	Description
tapiGetLocationInfo	This function returns the country code and city (area) code that the user has set in the current location parameters in the Telephony Control Panel. The application can use this information to assist the user in forming proper canonical telephone numbers, such as by offering these as defaults when new numbers are entered in a phone book entry or database record.
tapiRequestMakeCall	This function requests the establishment of a voice call. A call-manager application is responsible for establishing the call on behalf of the requesting application, which is then controlled by the user's call-manager application.
lineAddProvider	This function installs a telephony service provider.
lineAddToConference	This function adds the call specified by the hConsultCall parameter to the conference call specified by the hConfCall parameter.
lineAnswer	This function answers the specified offering call.
lineBlindTransfer	This function performs a blind or single-step transfer of the specified call to the specified destination address.
lineClose	This function closes the specified open line device.
lineCompleteTransfer	This function completes the transfer of the specified call to the party connected in the consultation call.
lineDrop	This function drops or disconnects the specified call. The application has the option to specify user-to-user data to be transmitted as part of the call disconnect.
1 of 3	

Table 21: Supported TAPI services (continued)

Service	Description
lineForward	This function forwards calls destined for the specified address on the specified line, according to the specified forwarding instructions. When an originating address (dsAddressID) is forwarded, the specified incoming calls for that address are deflected to the other number by the switch. This function provides a combination of forward and do-not-disturb features. This function can also cancel forwarding currently in effect.
lineGetAddressCaps	This function queries the specified address on the specified line device to determine its telephony capabilities.
lineGetAddressStatus	This function allows an application to query the specified address for its current status.
lineGetCallInfo	This function enables an application to obtain fixed information about the specified call.
lineGetCallStatus	This function returns the current status of the specified call.
lineGetDevCaps	This function queries a specified line device to determine its telephony capabilities. The returned data is valid for all addresses on the line device.
LineGetID	This function returns a device identifier for the specified device class associated with the selected line, address, or call.
lineGetLineDevStatus	This function enables an application to query the specified open line device for its current status.
lineHold	This function places the specified call on hold.
lineInitializeEx	This function initializes the application's use of TAPI for subsequent use of the line abstraction. It registers the application's specified notification mechanism and returns the number of line devices available to the application. A line device is any device that provides an implementation for the line-prefixed functions in the Telephony API.
lineMakeCall	This function places a call on the specified line to the specified destination address. Optionally, call parameters can be specified if anything but default call setup parameters are requested.
2 of 3	

Table 21: Supported TAPI services (continued)

Service	Description
lineNegotiateAPIVersion	This function enables an application to negotiate a TAPI version to use.
LineOpen	This function opens the line device specified by its device identifier and returns a line handle for the corresponding opened line device. This line handle is used in subsequent operations on the line device.
lineRedirect	This functions redirects the call to another line.
lineSetStatusMessages	This function enables an application to specify which notification messages the application wants to receive for events related to status changes for the specified line or any of its addresses.
lineSetupConference	This function sets up a conference call for the addition of the third party.
lineSetupTransfer	This function initiates a transfer of the call specified by the hCall parameter. It establishes a consultation call, lphConsultCall , on which the party can be dialed that can become the destination of the transfer. The application acquires owner privilege to the lphConsultCall parameter.
lineShutdown	This function shuts down the application's usage of the line abstraction of the API.
lineSwapHold	This function swaps the specified active call with the specified call on consultation hold.
lineTranslateAddress	This function translates the specified address into a dialable string format.
lineUnhold	This function retrieves the specified held call.
3 of 3	

TAPI-based applications can receive all the TAPI 2.1 and 2.2 events shown in [Table 22: Supported TAPI Events](#) with all data values populated.

Table 22: Supported TAPI Events

Event	Description	Constants Provided
LINE_CALLSTATE	Notifies the application that the status of the specified call has changed. Typically, several such messages are received during the lifetime of a call. Also informs the application of new incoming calls in the offering state. The application can use <code>lineGetCallStatus</code> to retrieve more information about the current status of the call.	LINECALLSTATE_OFFERING LINECALLSTATE_CONNECTED LINECALLSTATE_DIALING LINECALLSTATE_DIALTONE LINECALLSTATE_DICS_CONNECTED LINECALLSTATE_IDLE LINECALLSTATE_PROCEEDING LINECALLSTATE_RINGBACK LINECALLSTATE_ONHOLD LINECALLSTATE_BUSY LINECALLSTATE_CONFERENCED
LINE_CLOSE	Notifies the application that the specified line device has been forcibly closed. The line device handle or any call handles for call on the line are no longer valid.	
LINE_CREATE	Notifies the application of the creation of a new line device.	
LINE_LINEDEVSTATE	Notifies the application that the state of a line device has changed. The application can invoke <code>lineGetLineDevStatus</code> to learn the line's new status.	LINEDEVSTATE_CLOSE LINEDEVSTATE_CONNECTED LINEDEVSTATE_DISCONNECTED LINEDEVSTATE_REINIT LINEDEVSTATE_REMOVED LINEDEVSTATE_RINGING.
LINE_REMOVE	Notifies the application that a line device has been removed (deleted) from the system. Generally, this is not used for temporary removals, such as extraction of PCMCIA devices, but for permanent removals in which the device would no longer be reported by the service provider if TAPI were re-initialized.	
1 of 2		

Table 22: Supported TAPI Events (continued)

Event	Description	Constants Provided
LINE_CALLINFO	The TAPI LINE_CALLINFO message is sent when the call information about the specified call has changed. The application can invoke lineGetCallInfo to determine the current call information. <i>CallerID</i> (originating party of an incoming call) and <i>CalledID</i> are available immediately after the call is offered to the application. Applications should not necessarily wait for the LINE_CALLINFO message before trying to retrieve this information.	
LINE_REPLY	Informs the application about the results of function calls that were completed asynchronously.	

2 of 2

Supported data structure parameters

Parameters defined in the TAPI data structures are supported as indicated in the following Table. The following notation is used:

- -X - parameter is supported
- -NS - parameter is not supported

Reported via lineGetCallInfo request.

Table 23: Line Call Information

Field Name	Avaya TAPI TSP
dwCallerID	x
dwCallerIDFlags	LINECALLPARTYID_ADDRESS
dwCallerIDName	x (Switch Name)
dwCalledID	x

Table 23: Line Call Information

dwCalledIDFlags	LINECALLPARTYID_ADDRESS
dwCalledIDName	x (Switch name)
dwBearerMode	LINEBEARERMODE_VOICE
dwMediaMode	LINEMEDIAMODE_UNKNOWN LINEMEDIAMODE_INTERACTIVEVOICE
dwCallStates	LINECALLSTATE_IDLE LINECALLSTATE_DIALING LINECALLSTATE_RINGBACK LINECALLSTATE_CONNECTED LINECALLSTATE_PROCEEDING LINECALLSTATE_BUSY LINECALLSTATE_UNKNOWN
dwOrigin	LINECALLORIGIN_OUTBOUND LINECALLORIGIN_INBOUND
dwReason	LINECALLREASON_DIRECT
dwConnectedIDFlags	LINECALLPARTYID_UNAVAIL
dwRedirectionIDFlags	LINECALLPARTYID_UNAVAIL
dwRedirectingIDFlags	LINECALLPARTYID_UNAVAIL
dwDisplaySize	x
dwDisplayOffset	x
dwUserUserInfoSize	x
dwUserUserInfoOffset	x

Reported via lineGetDevCaps request.

Table 24: Line Dev Caps

Field Name	AvayaTSP30
dwProviderInfoSize	X
dwProviderInfoOffset	X
dwLineNameSize	X
dwLineNameOffset	X
dwStringFormat	STRINGFORMAT_ASCII

1 of 2

dwPermanentLineID	x
dwAddressModes	LINEADDRESSMODE_ADDRESSID
dwNumAddresses	x (1)
dwBearerModes	LINEBEARERMODE_VOICE LINEBEARERMODE_DATA LINEBEARERMODE_SPEECH
dwMediaModes	LINEMEDIAMODE_INTERACTIVEVOICE LINEMEDIAMODE_UNKNOWN LINEMEDIAMODE_DATAMODEM
dwGenerateDigitModes	LINEDIGITMODE_DTMF
dwDevCapFlags	LINEDEVCAPFLAGS_CROSSADDRCONF
dwAnswerMode	LINEANSWERMODE_HOLD
dwMaxNumActiveCalls	x (1)
dwLineFeatures	LINEFEATURE_MAKECALL
MinDialParams	x
MaxDialParams	x
DefaultDialParams	x
dwUUIAcceptSize	0
dwUUIAnswerSize	0
dwUUIMakeCallSize	ATT_MAX_UI_SIZE (96)
dwUIDropSize	ATT_MAX_UI_SIZE (96)
dwUISendUserUserInfoSize	0
dwUICallInfoSize	0

2 of 2

Used by lineMakeCall request

Table 25: Line Make Call Request

Field Name	Avaya TAPI TSP
dwUserUserInfoSize	X
dwUserUserInfoOffset	X

Used by lineSetupTransfer request.

Table 26: Line Setup Transfer Request

Field Name	Avaya TAPI TSP
dwUserUserInfoSize	X
dwUserUserInfoOffset	X

Used by lineSetupConference request.

Table 27: Line Setup Conference

Field Name	Avaya TAPI TSP
dwUserUserInfoSize	X
dwUserUserInfoOffset	X

Reported by lineGetAddressCaps request.

Table 28: Line Get Address Caps

Field Name	Avaya TAPI TSP
dwLineDeviceID	X
dwAddressSize	X
dwAddressOffset	NS
dwDevSpecificSize	NS
dwDevSpecificOffset	NS
dwAddressSharing	LINEADDRESSSHARING_PRIVATE
dwAddressStates	NS
dwCallInfoStates	LINECALLINFOSTATE_MEDIAMODE LINECALLINFOSTATE_APPSPECIFIC
dwCallerIDFlags	LINECALLPARTYID_ADDRESS LINECALLPARTYID_UNAVAIL
dwCalledIDFlags	LINECALLPARTYID_ADDRESS LINECALLPARTYID_UNAVAIL
dwConnectedIDFlags	
dwRedirectionIDFlags	LINECALLPARTYID_UNAVAIL

1 of 4

Table 28: Line Get Address Caps

dwRedirectingIDFlags	LINECALLPARTYID_UNAVAIL
dwCallStates	LINECALLSTATE_IDLE LINECALLSTATE_DIALTONE LINECALLSTATE_DIALING LINECALLSTATE_CONNECTED LINECALLSTATE_ACCEPTED LINECALLSTATE_OFFERING LINECALLSTATE_ONHOLD LINECALLSTATE_RINGBACK LINECALLSTATE_DIALTONE LINECALLSTATE_ONHOLDPENDTRANSFER
dwDialToneModes	LINEDIALTONEMODE_UNAVAIL
dwBusyModes	LINEBUSYMODE_UNAVAIL
dwSpecialInfo	LINESPECIALINFO_UNAVAIL
dwDisconnectModes	LINEDISCONNECTMODE_UNAVAIL
dwMaxNumActiveCalls	1
dwMaxNumOnHoldCalls	64
dwMaxNumOnHoldPending Calls	0
dwMaxNumConference	0
dwMaxNumTransConf	0
dwAddrCapFlags	LINEADDRCAPFLAGS_DIALED
dwCallFeatures	LINECALLFEATURE_DIAL LINECALLFEATURE_DROP LINECALLFEATURE_HOLD LINECALLFEATURE_GENERATEDIGITS LINECALLFEATURE_ANSWER LINECALLFEATURE_BLINDTRANSFER LINECALLFEATURE_COMPLETETRANSF LINECALLFEATURE_UNHOLD LINECALLFEATURE_SWAPHOLD LINECALLFEATURE_SETUPTRANSFER LINECALLFEATURE_SETUPCONF LINECALLFEATURE_ADDTOCONF LINECALLFEATURE_REDIRECT

2 of 4

Table 28: Line Get Address Caps

dwRemoveFromConfCaps	NS
dwRemoveFromConfState	NS
dwTransferModes	LINETRANSFERMODE_TRANSFER LINETRANSFERMODE_CONFERERENCE
dwParkModes	NS
dwForwardModes	NS
dwMaxForwardEntries	NS
dwMaxSpecificEntries	NS
dwMinFwdNumRings	NS
dwMaxFwdNumRings	NS
dwMaxCallCompletions	NS
dwCallCompletionConds	NS
dwCallCompletionModes	NS
dwNumCompletionMessages	NS
dwCompletionMsgTextEntry Size	NS
dwCompletionMsgTextSize	NS
dwCompletionMsgTextOffset	NS
dwAddressFeatures	LINEADDRFEATURE_MAKECALL LINEADDRFEATURE_PICKUP
dwPredictiveAutoTransferSta tes	NS
dwNumCallTreatments	NS
dwCallTreatmentListSize	NS
dwCallTreatmentListOffset	NS
dwDeviceClassesSize	NS
dwDeviceClassesOffset	NS
dwMaxCallDataSize	NS

3 of 4

Table 28: Line Get Address Caps

dwCallFeatures2	NS
dwMaxNoAnswerTimeout	NS
dwConnectedModes	NS
dwOfferingModes	NS
dwAvailableMediaModes	NS
4 of 4	

Reported by lineGetAddressStatus request.

Table 29: Line Get Address Status

Field Name	Avaya TAPI TSP
dwNumInUse	NS
dwNumActiveCalls	X
dwNumOnHoldCalls	X
dwNumOnHoldPendCalls	X
dwAddressFeatures	LINEADDRFEATURE_MAKECALL
dwNumRingsNoAnswer	NS
dwForwardNumEntries	NS
dwForwardSize	NS
dwForwardOffset	NS
dwTerminalModesSize	NS
dwTerminalModesOffset	NS
dwDevSpecificSize	NS
dwDevSpecificOffset	NS

Reported by lineGetCallStatus request

Table 30: Line Get Status

Field Name	Avaya TAPI TSP
dwCallState	LINECALLSTATE_IDLE LINECALLSTATE_CONNECTED LINECALLSTATE_OFFERING LINECALLSTATE_ONHOLD LINECALLSTATE_DIALTONE
dwCallStateMode	NS
dwCallPrivilege	NS
dwCallFeatures	LINECALLSTATE_IDLE 0 LINECALLSTATE_CONNECTED LINECALLFEATURE_DROP LINECALLFEATURE_HOLD LINECALLFEATURE_SWAPHOLD LINECALLFEATURE_SETUPTRANSFER LINECALLFEATURE_COMPLETETRANSF LINECALLFEATURE_SETUPCONF LINECALLFEATURE_BLINDTRANSFER LINECALLFEATURE_ADDTOCONF LINECALLSTATE_OFFERING LINECALLFEATURE_ANSWER LINECALLFEATURE_REDIRECT LINECALLSTATE_ONHOLD LINECALLFEATURE_COMPLETETRANSF LINECALLFEATURE_SWAPHOLD LINECALLFEATURE_UNHOLD LINECALLFEATURE_DROP LINECALLSTATE_DIALTONE LINECALLFEATURE_DIAL LINECALLFEATURE_DROP LINECALLFEATURE_SETUPTRANSFER LINECALLFEATURE_COMPLETETRANSF
dwDevSpecificSize	NS

1 of 2

Table 30: Line Get Status

dwDevSpecificOffset	NS
dwCallFeatures2	NS
tStateEntryTime	NS
tStateEntryTime	NS
2 of 2	

Supported TAPI Services and Events

Reported by lineGetLineDevStatus request.

Table 31: Line Get Line Development Status

Field Name	Avaya TAPI TSP
dwNumOpens	NS
dwOpenMediaModes	NS
dwNumActiveCalls	X
dwNumOnHoldCalls	X
dwNumOnHoldPendCalls	X
dwLineFeatures	NS
dwNumCallCompletions	NS
dwRingMode	NS
dwSignalLevel	NS
dwBatteryLevel	NS
dwRoamMode	NS
dwDevStatusFlags	LINEDEVSTATUSFLAGS_CONNECTED LINEDEVSTATUSFLAGS_INSERVICE
dwTerminalModesSize	NS
dwTerminalModesOffset	NS
dwDevSpecificSize	NS
dwDevSpecificOffset	NS
dwAvailableMediaModes	NS
dwAppInfoSize	NS
dwAppInfoOffset	NS

Appendix C: CTI Programming Considerations, Design Tips, and Best Practices

In call flow scenarios observed through CTI, changes in the states of devices, calls, connections and features are reported through CTI events provided over CTI monitors. The flow of events can be complicated and not easily predicted, especially when multiple devices are involved in the scenario - a typical scenario when lines are shared among several devices or when conference or transfer features are used. In addition, events indicating state connection transitions at other devices than a monitored device, may not be provided over the monitored device's monitor¹.

Therefore, it is a strongly advised programming recommendation that CTI applications should always be capable of gracefully handling unexpected CTI events and event parameter values (particularly, **event Cause parameter** values.)

As a general rule, when system behaviors resulting from features that involve multiple devices are invoked or observed through CTI, the most complete and informative flow of CTI events results from CTI monitoring of all the involved devices. For example, when a call arrives on a line that is shared by several devices, the most complete and informative flow of CTI events can be achieved when CTI monitors have been established at all the devices that share the line. In the cases of Collective Ringing Groups (CRGs) and Outside Line Groups (OLGs), it is recommended that all the phones comprising such groups be monitored in order to achieve the more complete view of the state of any device or call in the Group.

In the following sections examples of CTI event flows and system behavior involving lines/line appearances, CRGs and OLGs are given. These may show situations in which the above recommendations can be useful to a CTI-enabled application. These examples are provided as examples of the kinds of scenarios and resulting events that can arise with CTI control/monitoring involving key-system features. They illustrate types of implementation-specific behaviors that should be considered when porting, developing and/or debugging CTI applications in these configurations. However, the set of scenarios is not exhaustive and is provided only as illustrative guidance.

1. For example, if devices A and B are members of the same Outgoing Line Group (i.e., both have Line Appearances of the same outside line) and a CTI monitor is established on A, then when an incoming call arrives both A and B will alert and a Delivered Event will flow over A's monitor. But if B answers the call (transitioning from Alerting to Connected state), an Established event (indicating B's connection transition) will not flow over A's CTI monitor. If an application established a monitor on B, as well as A, then the application would be correctly informed of the connection state transition at B.

Collective Ringing Groups

The following call flows and behaviors may be observed in scenarios involving Collective Ringing Groups (CRGs). It is recommended that CTI monitors be established on all member phone devices in a CRG whose comprising devices are to be observed and/or controlled through CTI.

Incoming call to CRG at a monitored member station that answers

This is a straightforward scenario. It is provided mainly to illustrate the cause values provided in the events.

(Assuming no other monitored stations in the group)

Delivered Event

Cause: Call_FORWARD_BUSY

Station answers the call

Established Event

Cause: KEY_CONFERENCE

Station clears the call

Connection Cleared Event

Cause: KEY_CONFERENCE

Incoming call to CRG at a monitored member station that is answered elsewhere

This scenario is similar to the previous one but illustrates events at the non-answering station. (The previous scenario illustrated events at the answering station.) Note that the call moves away from the non-answering station when the other station answers the call and this is conveyed via a Diverted event.

(Assuming no other monitored stations in the group)

Delivered Event

Cause: Call_FORWARD_BUSY

Another station answers the call

Diverted Event

Cause: REDIRECTED

Answering station clears the call. No further events result.

Incoming call to CRG at a monitored member station that is dropped before answer

In this scenario the calling party disconnects before the call is answered. The scenario is provided to illustrate the cause values provided in the events and to advise that additional Connection Cleared events will be received for all group members with lower extension numbers than that of the one being monitored.

(Assuming no other monitored stations in the group)

Delivered Event

Cause: Call_FORWARD_BUSY

Calling party drops while alerting

Connection Cleared Event

Connection Cleared Event

Connection Cleared Event

Cause: KEY_CONFERENCE

Outside Line Group Feature Behavior

An Outside Line Group (OLG) is a group of stations that share (i.e., have Line Appearances for) the same outside direct line. The following call flows and behaviors may be observed in scenarios involving OLGs. It is recommended that when using CTI to observe call activity on OLG member stations, CTI monitors be established on all phone devices that are members of that OLG.

Incoming call on OLG line

In this scenario an incoming call on an outside line arrives and is presented to two stations, A and B, with line appearances on the line (i.e. the stations are in the same OLG). Both stations are monitored. The call is answered by Station B (but not A). An Established event on station A's monitor is received when station B answers. But the event on A's monitor does not show Station B's device ID.

On device A's monitor (call is not answered at this device):

Established Event [Status: 265127496 or ID Not Known Type: Device Identifier Device ID: (null)]

On device B's monitor (where the call was actually answered):

Established Event: [Status: ID Provided Type: Explicit Private Local Number Device ID: 207]

Incoming call to OLG is answered by one station and other station connections transition to null

In this scenario, an incoming call to an OLG is answered by one of the OLG members. Events arising at other stations in the OLG or snapshot queries of these other stations will show local connection information indicating that the connection state to the answered call is Null. In fact, the other station's LED associated with the outside line will indicate that the line is in use, so the Null connection information is somewhat misleading. The other stations will receive a Cleared Connection event when the station that answered the call drops from it.

The same Null connection state will be seen if one OLG member is active on a call with other OLG member stations and then hangs up. In this case, the local connection state will be reported as Null.

localConnInfo: Null - when OLG group member was alerting and another member answered the call, or is on a call with other members and hangs up, goes to Null state.

Incoming answered call to monitored OLG member on Line Appearance and additional OLG members subsequently join call

In this scenario, the value of the Called Device parameter may not be consistent among Established events that arise as additional OLG members join the call. In this scenario monitors are active on each of three OLG members. As each station joins the call, an Established event is provided over the station's monitor. In this example, extension 270 answers the incoming call. Non-answering station extension 440 (as well as answering extension 270) will receive an Established event.

In the Established event at the non-answering extension 440, the Called Device is reported as the answering extension 270 and the Answering Device is reported as the non-answering station 440.

Established Event (at non-answering station extension 270):**Est Conn:** [Call ID: 2 Static Device ID: 440]**Answering Device:** [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]**Calling Device:** [Status: ID Provided Type: Explicit Public Unknown Device ID: T2#1]**Called Device:** [Status: ID Provided Type: Explicit Private Local Number Device ID: 207]**Last Redir Device:** [Status: ID Not Required]**Local Conn Info:** NullCause: NEW_CALL

Now the originally non-answering station extension 440 also answers the call. Both extensions 440 (the now-answering extension) and extension 270 (the originally-answering extension) receive Established events. On the monitor of the now-answering station 440 (Monitor Cross Reference 2, e.g.) the now-answering station shows its own extension as the Called Device ID:

Established Event at now-answering station (440):**Est Conn:** [Call ID: 2 Static Device ID: 440]**Answering Device:** [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]**Calling Device:** [Status: ID Provided Type: Explicit Public Unknown Device ID: T2#1]**Called Device:** [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]**Last Redir Device:** [Status: ID Not Required]**Local Conn Info:** ConnectedCause: KEY_CONFERENCE

In the Established event at the originally-answering device (x270) (reported over Monitor Cross Reference 1) the originally-answering party also shows the now-answering station extension (440) as the Called Device:

Established: Event at originally-answering device (270):**Est Conn:** [Call ID:2 Static Device ID: 440]**Answering Device:** [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]**Calling Device:** [Status: ID Provided Type: Explicit Public Unknown Device ID: T2#1]**Called Device:** [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]**Last Redir Device:** [Status: ID Not Required]**Local Conn Info:** ConnectedCause: KEY_CONFERENCE

Note that the Called Device values in both Established Events arising when Station 440 answers are in agreement. But they disagree with the Called Device value in the Established Event that occurred when the first station (x270) answered.

Incoming, answered call to monitored OLG member on Line Appearance and additional OLG members subsequently join call

In this scenario, the value of the Called Device parameter may not be consistent among Established events that arise as additional OLG members join the call. In this scenario monitors are active on each of three OLG members. As each station joins the call, an Established event is provided over the station's monitor. In this example, extension 270 answers the incoming call. Non-answering station extension 440 (as well as answering extension 270) will receive an Established event.

In the Established event at the non-answering extension 440, the Called Device is reported as the answering extension 270 and the Answering Device is reported as the non-answering station 440.

Established Event (at non-answering station extension 270):

Est Conn: [Call ID: 2 Static Device ID: 440]

Answering Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Calling Device: [Status: ID Provided Type: Explicit Public Unknown Device ID: T2#1]

Called Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 207]

Last Redir Device: [Status: ID Not Required]

Local Conn Info: NullCause: NEW_CALL

Now the originally non-answering station extension 440 also answers the call. Both extensions 440 (the now-answering extension) and extension 270 (the originally-answering extension) receive Established events. On the monitor of the now-answering station 440 (Monitor Cross Reference 2, e.g.) the now-answering station shows its own extension as the Called Device ID:

Established Event at now-answering station (440):

Est Conn: [Call ID: 2 Static Device ID: 440]

Answering Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Calling Device: [Status: ID Provided Type: Explicit Public Unknown Device ID: T2#1]

Called Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Last Redir Device: [Status: ID Not Required]

Local Conn Info: ConnectedCause: KEY_CONFERENCE

In the Established event at the originally-answering device (x270) (reported over Monitor Cross Reference 1) the originally-answering party also shows the now-answering station extension (440) as the Called Device:

Established: Event at originally-answering device (270):

Est Conn: [Call ID:2 Static Device ID: 440]

Answering Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Calling Device: [Status: ID Provided Type: Explicit Public Unknown Device ID: T2#1]

Called Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Last Redir Device: [Status: ID Not Required]

Local Conn Info: ConnectedCause: KEY_CONFERENCE

Note that the Called Device values in both Established Events arising when Station 440 answers are in agreement. But they disagree with the Called Device value in the Established Event that occurred when the first station (x270) answered.

Second OLG station to join OLG call hangs up. The first monitored station that originally answered (still on the call) gets TWO Connection Cleared events

In this scenario the second (i.e., the station that did not answer or make the call) of two OLG stations conferenced with another party on the line, hangs up. The remaining OLG station remaining on the call receives two connection cleared events. Note the difference in the Cause parameters of NONE vs. KEY_CONFERENCE. This (two Connection Cleared events) happens again later when the entire call goes away and all the remaining stations get an extra event showing themselves clearing connection for the call.

(First) Connection Cleared Event:

Dropped Conn: [Call ID: 2 Static Device ID: 440]

Releasing Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Local Conn Info: Connected

Cause: NONE

Private Data:

Vendor: ECS

Event Type :ATT_CONNECTION_CLEARED

Service: Connection Cleared

UUI Type: None

Number of DeviceHistory Entries: 1

Old Device ID: 440

Cause: NONE

Old Connection ID[Call ID: 2 Static Device ID: 440]

(Second) Connection Cleared Event:

Dropped Conn: [Call ID: 2 Static Device ID: 440]

Releasing Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Local Conn Info: Connected

Cause: KEY_CONFERENCE

Private Data:

Vendor: ECS

Event Type :ATT_CONNECTION_CLEARED

Service: Connection Cleared

UUI Type: None

Number of DeviceHistory Entries: 1

Old Device ID: 440

Cause: KEY_CONFERENCE

Old Connection ID: [Call ID: 2 Static Device ID: 440]

Snapshot call does not always show all devices: example, when an OLG member hangs up and the call continues with other OLG member(s)

Snapshot Call CONFIRM:Snapshot Call:

Number of Devices: 3

Device: [Status: ID Provided Type: Explicit Public Unknown Device ID: T2#1]

Call: [Call ID: 2 Dynamic Device ID: T2#1]

Local Conn Info: Connected

Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 207]

Call: [Call ID: 2 Static Device ID: 207]

Local Conn Info: Connected

Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 500]

Call: [Call ID: 2 Static Device ID: 500]

Local Conn Info: Null

Private Data:

Vendor: ECS

Event Type :ATT_SNAPSHOT_CALL_CONF

Service: Snapshot Call

Number of DeviceHistory Entries: 1

Old Device ID 440

Cause: KEY_CONFERENCE

Old Connection ID: [Call ID: 2 Static Device ID: 440]

After snapshotting each device, however, snapshot call will show all devices:

CONFIRM:Snapshot Call:

Number of Devices: 4

Device: [Status: ID Provided Type: Explicit Public Unknown Device ID: T2#1]

Call: [Call ID: 2 Dynamic Device ID: T2#1]

Local Conn Info: Connected

Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 207]

Call: [Call ID: 2 Static Device ID: 207]

Local Conn Info: Connected

Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 500]

Call: [Call ID: 2 Static Device ID: 500]

Local Conn Info: Null

Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Call: [Call ID: 2 Static Device ID: 440]

Local Conn Info: Null

Private Data:

Vendor: ECS

Event Type: ATT_SNAPSHOT_CALL_CONF

Service: Snapshot Call

Number of DeviceHistory Entries: 1

Old Device ID 440

Cause: KEY_CONFERENCE

Old Connection ID: [Call ID: 2 Static Device ID: 440]

Last OLG Station Clears From Call - Multiple Connection Cleared Events Received

In this example, three stations are in the same OLG and were on the same call (Does this happen if they were not active on the call?) but two stations have hung up. The final station (extension 440 in this example) now hangs up. Three Connection Cleared events are received at the last-clearing OLG member station as the call is cleared from each OLG member.

Connection Cleared Event:

Dropped Conn: [Call ID: 2 Static Device ID: 440]

Releasing Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Local Conn Info: Null

Cause: KEY_CONFERENCE

Private Data:

Vendor: ECS

Event Type: ATT_CONNECTION_CLEARED

Service: Connection Cleared

UUI Type: None

Number of Device History Entries: 1

Old Device ID 440

Cause: KEY_CONFERENCE

Old Connection ID: [Call ID: 2 Static Device ID: 440]

Connection Cleared Event:

Dropped Conn: [Call ID: 2 Static Device ID: 440]

Releasing Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Local Conn Info: Null

Cause: KEY_CONFERENCE

Private Data:

- **Vendor:** ECS
- **Event Type :**ATT_CONNECTION_CLEARED
- **Service:** Connection Cleared
- **UUI Type:** None
- **Number of Device History Entries:** 1
- **Old Device ID** 440
- **Cause:** KEY_CONFERENCE
- **Old Connection ID:** [Call ID: 2 Static Device ID: 440]

Connection Cleared Event:

Dropped Conn: [Call ID: 2 Static Device ID: 440]

Releasing Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Local Conn Info: Null

Cause: KEY_CONFERENCE

A CTI-Extension that is a member of an OLG cannot answer/join an OLG call that is already active at one or more other "real" members of the same OLG

An Answer Call service request is issued on behalf of a CTI Extension, requesting that it be joined to a call that is present on its outside line. Other stations in the same OLG are already active on the call. The request to add the CTI Extension will fail. Note that a request to add a "real" physical phone would be allowed.

Answer Call service request:

Alerting Call: [Call ID: 11 Static Device ID: 500]

Answer Call request is acknowledged:

CONFIRM:

Answer Call request is denied as indicated by universal failure event:

Universal Failure:

Error: REQUEST_TIMEOUT_REJECTION

Or, depending on exact processing timing, an alternative error reason code may be provided in the universal failure event:

Universal Failure:

Error: RESOURCE_BUSY

No Connection Cleared Event from Outgoing OLG Call

In this scenario, a monitored OLG member manually (i.e. not via CTI) places an outgoing (external) call that is answered by the called party. When the call is placed the call ID of the new call is reported in the Initiated event. When the external called party disconnects (thus ending the call), the end of the call (and its clearing from the monitored device) does not generate a Connection Cleared event. (Note that a Connection Cleared event is generated when the call is placed via a Call Appearance, rather than an OLG Line Appearance.) However, if a call monitor

was placed on the call (using the CallID reported in the Initiated event), then the application will receive a Call Cleared event to signal the clearing of the call. Therefore, it is recommended that call monitors be established when an OLG member places a call via its OLG Line Appearance.

Multiple Conferenced and Transferred events

When CTI is used to conference or transfer a call present on an OLG member's Line Appearance, two Conferenced (or Transferred) events may be seen over the conferencing device's (OLG member's) device monitor. If the call is present on a Call Appearance (rather than Line Appearance) only a single Conferenced (or Transferred) event will be received. The dual Conferenced event behavior is shown in the following example. In this case, CTI Conference service is invoked on behalf of an OLG member station (conferencing device). A to-be-conferenced call on the OLG Line Appearance is on hold and a consultation call is active with another station (to be added to the conference). Be aware that the reported local connection states reported in the two Conferenced events (Held and Null in the following example) differ from those reported when the conferencing station is not at OLG member conferencing a call on the OLG's Line Appearance (in which case the reported local call state at the conferencing device will be Connected.)

Conference Call service request:

Held Call: [Call ID: 86 Static Device ID: 440]

Active Call: [Call ID: 87 Static Device ID: 440]

First Conferenced event:

Primary Old Call: [Call ID: 86 Static Device ID: 440]

Secondary Old Call: [Call ID: 87 Static Device ID: 440]

Conf Controller: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Added Party: [Status: ID Provided Type: Explicit Private Local Number Device ID: 230]

Conference Connections:

Number of Connections: 3

Party: [Call ID: 87 Static Device ID: 440]

Static Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Party: [Call ID: 87 Static Device ID: 230]

Static Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 230]

Party: [Call ID: 87 Dynamic Device ID: T86#1]

Static Device: [Status: ID Provided Type: Explicit Public Unknown Device ID: T86#1]

Local Conn Info: Hold

Cause: NONE

Private Data:Vendor: ECS

Event Type :ATT_CONFERENCED

Service: Conferenced

Orig Call Info Reason: Conferenced

OCI Calling Device: [Status: ID Provided Type: Explicit Public Unknown Device ID: T86#1]

OCI Called Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

OCI Trunk Group: 2OCI Trunk Member: (null)

OCI Look Ahead Interflow Type: No Interflow

OCI LAI Priority: Not In Queue

OCI LAI Hours: 0, Minutes: 0, Seconds: 0

OCI LAI Source VDN: (null)

OCI LAI Unicode Source VDN:

OCI User Entered Code Type: None

OCI UEC Indicator: CollectOCI UEC Data: (null)

OCI UEC Collect VDN: (null)

OCI UII Type: None

OCI Universal Call ID: 00000000000000000000

OCI Call Originator Type: NO INFO

OCI Flexible Billing: NO

OCI Device History: (null)

Distributing Device: [Status: ID Not Known]

Universal Call ID: 00000000000000000000

Conference Trunk List:Number of Trunk Connections: 1

Trunk Info: [Call ID: 87 Dynamic Device ID: T86#1]

Trunk Group: 2

Trunk Member:

Device History: (null)

Distributing VDN: [Status: ID Not Known]

Conference Call service request CONFIRM: Conference Call:

New Call: [Call ID: 87 Static Device ID: 440]

Conn List:

Number of Connections: 3

Party: [Call ID: 87 Static Device ID: 440]

Static Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Party: [Call ID: 87 Static Device ID: 230]

Static Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 230]

Party: [Call ID: 87 Dynamic Device ID: T86#1]

Static Device: [Status: ID Provided Type: Explicit Public Unknown Device ID: T86#1]

Private Data:Vendor: ECS

Event Type :ATT_CONFERENCE_CALL_CONF

Service: Conference Call

UCID: 00000000000000000000

SECOND Conferenced Event:

Primary Old Call: [Call ID: 86 Static Device ID: 440]

Secondary Old Call: [Call ID: 87 Static Device ID: 440]

Conf Controller: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Added Party: [Status: ID Provided Type: Explicit Private Local Number Device ID: 230]

Conference Connections:Number of Connections: 3

Party: [Call ID: 87 Static Device ID: 440]

Static Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

Party: [Call ID: 87 Static Device ID: 230]

Static Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 230]

Party: [Call ID: 87 Dynamic Device ID: T86#1]

Static Device: [Status: ID Provided Type: Explicit Public Unknown Device ID: T86#1]

Local Conn Info: None

Cause: NONE

Private Data:Vendor: ECS

Event Type :ATT_CONFERENCED

Service: Conferenced

Orig Call Info Reason: Conferenced

OCI Calling Device: [Status: ID Provided Type: Explicit Public Unknown Device ID: T86#1]

OCI Called Device: [Status: ID Provided Type: Explicit Private Local Number Device ID: 440]

OCI Trunk Group: 2OCI Trunk Member: (null)

OCI Look Ahead Interflow Type: No Interflow

OCI LAI Priority: Not In Queue

OCI LAI Hours: 0, Minutes: 0, Seconds: 0

OCI LAI Source VDN: (null)

OCI LAI Unicode Source VDN:

OCI User Entered Code Type: None

OCI UEC Indicator: CollectOCI UEC Data: (null)

OCI UEC Collect VDN: (null)

OCI UII Type: None

OCI Universal Call ID: 00000000000000000000

OCI Call Originator Type: NO INFO

OCI Flexible Billing: NO

OCI Device History: (null)

Distributing Device: [Status: ID Not Known]

Universal Call ID: 00000000000000000000

Conference Trunk List:Number of Trunk Connections: 1

Trunk Info: [Call ID: 87 Dynamic Device ID: T86#1]

Trunk Group: 2

Trunk Member:

Device History: (null)

Distributing VDN: [Status: ID Not Known]

Appendix D: About the Avaya DevConnect Program

The Avaya DevConnect program provides a wide range of developer oriented resources, including access to APIs and SDK downloads and documentation, developer tools, sample applications, technical support options and training resources. Access to these resources requires an authenticated login ID and password.

Basic DevConnect membership is free and open to all Avaya customers and partners. Enhanced membership options include increased levels of technical developer support, compliance testing, procurement discounts and joint marketing benefits.

For more information, visit www.avaya.com/devconnect.

Glossary

A

API	Application Programming Interface A set of routines, protocols, and tools for building software applications.
application computer	For CTI , the computer that runs the TAPI-based application and the Avaya Application Enablement Services TAPI Telephony Service Provider .
Avaya Application Enablement Services TAPI Telephony Service Provider	A complete set of CTI software components that expose Communication Manager Branch CTI services to TAPI over the Avaya TAPI TSP interface. This software allows TAPI-based applications to run on an application computer connected to . Also referred to in this book as the <i>Avaya TAPI TSP</i> .
Avaya TAPI TSP	See Avaya Application Enablement Services TAPI Telephony Service Provider

C

Communication Manager Branch	A highly-distributed solution for branch offices that develops telephony and communications applications at a low cost. Communication Manager Branch can be used in individual locations, as well as in corporate enterprises with up to 1000 satellite locations.
Communication Manager Branch application enablement	A feature of that provides Computer Telephony Integration (CTI) for basic telephony at any Communication Manager Branch location.
CTI	Computer Telephony Integration A technology that allows interactions on a telephone and a computer to be integrated or coordinated. As contact channels have expanded from voice to include E-mail, web, and fax, the definition of CTI has expanded to include the integration of all customer contact channels (voice, E-mail, web, fax, etc.) with computer systems
CTI user	An external computer that connects to Communication Manager Branch via a LAN to receive CTI services. Each CTI user provides a CTI user ID (or name) when connecting to Communication Manager Branch, and this is used by Communication Manager Branch to authenticate the "identity" of computers that attempt to establish CTI sessions.

S

SIP	Session Initiation Protocol.
------------	------------------------------

Contents

An application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution, and multimedia conferences.

T

TAPI

Telephony Application Programming Language.

A Microsoft-created API that enables computers using Microsoft Windows to use telephone services.

TAPI Service Provider

A driver that allows TAPI-based applications to communicate with different types of TAPI hardware.

TSAPI

Telephony Services Application Programming Language.

An Avaya-proprietary API that integrates server-based telephony control with desktop or server applications on enterprise-wide networks. Although you may see references to TSAPI in some of the Communication Manager Branch application enablement components, TSAPI is not a supported API for Communication Manager Branch application enablement.

TSP

See [TAPI Service Provider](#)

V

VoIP

Voice over IP.

A set of facilities that use the Internet Protocol (IP) to manage the delivery of voice information. In general, VoIP means to send voice information in digital form in discrete packets instead of in the traditional circuit-committed protocols of the public switched telephone network (PSTN).

Index

A

API [121](#)
 authentication
 Local CTI user [17](#)
 Windows Active Directory Services [18](#)
 Avaya TAPI Service Provider, see TAPI Service Provider (TSP)
 Avaya TSP [11](#)

B

Branch Device Manager screens
 CTI Configuration screen [22](#), [45](#)
 CTI Maintenance screen [46](#)

C

configurations
 supported [7](#)
 CTI Configuration screen [22](#), [45](#)
 CTI Maintenance screen [46](#)

D

Domain Controller [20](#)

L

Local CTI user authentication [17](#)
 log file for errors and exceptions [45](#)

S

screens
 CTI Configuration screen [22](#), [45](#)
 CTI Maintenance screen [46](#)
 SIP endpoints, no support [14](#)
 supported configurations [7](#)
 supported TAPI versions [12](#)

T

TAPI
 supported versions [12](#)
 TAPI Service Provider (TSP)
 explanation

TSP, see TAPI Service Provider (TSP)

V

VOIP [122](#)

W

Windows Active Directory Services authentication . . . [18](#)

