Lucent Technologies
Bell Labs Innovations

PassageWay® Telephony Services for Windows NT® Release 2.31

Java™ Telephony API (JTAPI) Client Programmer's Guide Issue 1.0 October 1997

Notice

Every effort was made to ensure that the information in this book was complete and accurate at the time of printing. However, information is subject to change.

Your Responsibility for Your System's Security

Toll fraud is the unauthorized use of your telecommunications system by an unauthorized party, for example, persons other than your company's employees, agents, subcontractors, or persons working on your company's behalf. Note that there may be a risk of toll fraud associated with your telecommunications system and, if toll fraud occurs, it can result in substantial additional charges for your telecommunications services.

You and your system manager are responsible for the security of your system, such as programming and configuring your equipment to prevent unauthorized use. The system manager is also responsible for reading all installation, instruction, and system administration documents provided with this product in order to fully understand the features that can introduce risk of toll fraud and the steps that can be taken to reduce that risk. Lucent Technologies does not warrant that this product is immune from or will prevent unauthorized use of common-carrier telecommunication services or facilities accessed through or connected to it. Lucent Technologies will not be responsible for any charges that result from such unauthorized use.

Lucent Technologies Fraud Intervention

If you *suspect that you are being victimized* by toll fraud and you need technical support or assistance, call Technical Service Center Toll Fraud Intervention Hotline at **1 800 643 2353**.

Obtaining Products

To learn more about Lucent Technologies products and to order products, contact Lucent Direct, the direct-market organization of Lucent Technologies Business Communications Systems. Access their web site at **www.lucentdirect.com**. Or call the following numbers: customers **1 800 451 2100**, account executives **1 800 778 1881** (fax) or **1 800 778 1880** (voice).

Trademarks

Adobe, Acrobat, and the Acrobat logo are trademarks of Adobe Systems Incorporated, which may be registered in certain jurisdictions.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

PassageWay and the Lucent Technologies logotype are registered trademarks of Lucent Technologies Incorporated.

Windows NT is a registered trademark of Microsoft Corp.

All products and company names are trademarks or registered trademarks of their respective holders.

Comments

If you have comments, complete and return the comment card at the end of this document.

Acknowledgment

This document was prepared by BCS Product Publications, Lucent Technologies, Middletown, NJ 07748-9972.

Contents

About This Guide

Contents

About This Guide

What is JTAPI?

The Java™ Telephony API (JTAPI) specifies the standard telephony application programming interface for computer-telephone applications under Java. It is the definition for a reuseable set of call control objects that bring cross-platform and cross-implementation portability to telephony applications. It is a simple, extensible, object-oriented model that addresses a broad range of computer-telephony tasks.

The Java Telephony API represents the combined efforts of design teams from Sun, Lucent Technologies, Nortel, Novell, Intel, and IBM, all operating under the direction of JavaSoft.

What is the PassageWay Telephony Services Java Client?

The JTAPI specification, as published under the direction of JavaSoft, is a set of Java interfaces. It requires an implementation of the defined interfaces in the form of Java classes in order to produce a working product. The Lucent Technologies PassageWay Telephony Services Java client is an implementation of the Java classes required to support the JTAPI interfaces.

The Telephony Services Java client communicates with a Telephony Services driver (i.e., a service provider) to execute the telephony requests to control the actual switching elements. The Telephony Services drivers use Lucent Technologies Telephony Services Application Programmer's Interface (TSAPI) interfaces to present their services. Therefore, the Telephony Services Java client can only present the functions and services supplied by TSAPI and TSAPI's associated private data.

Since the Telephony Services Java client uses TSAPI as its vehicle to communicate with the Telephony Services drivers, the Telephony Services Java client has extended JTAPI to give the application additional information (provided by TSAPI) to help understand exceptions in greater detail. In addition to the parameters specified in the JTAPI protocol, the Telephony Services Java client can also deliver private data to the application that is unique to each switching environment. This additional information is provided by the switch providers in the private data that accompanies TSAPI messages.

Included in the Telephony Services Java client is a set of Java interfaces that gives the application programmer access to the private data information used by the DEFINITY Enterprise Communications Server (ECS). This allows the application programmer to access additional switch features not available through standard JTAPI. Application programmers who want their applications to run on multiple TSAPI switches must "special-case" their code using these extensions.

For switches other than DEFINITY, the Telephony Services Java client implements the JTAPI private data package. This exposes TSAPI private data directly to the application. Additional interfaces have been defined to allow the application programmer to translate from TSAPI constructs to JTAPI objects and allow the interpretation of private TSAPI information to JTAPI objects.

Purpose and Scope

This document describes:

- the Lucent Technologies generic implementation of JTAPI on PassageWay Telephony Services. This implementation provides a programming environment that may be used with any switch for which there is a PassageWay Telephony Services driver.

- the Lucent Technologies PassageWay Telephony Services[1] implementation of JTAPI that provides Telephony Services extensions to JTAPI exceptions for those application programmers who want to use TSAPI-specific error codes. This implementation provides a programming environment that may be used with any switch for which there is a PassageWay Telephony Services driver.

- the Telephony Services implementation of JTAPI that applies to clients using the DEFINITY switch and the associated PassageWay Telephony Services driver, the G3 PBX driver (G3PD). This implementation provides a programming environment that makes available DEFINITY-specific features.

- the Telephony Services implementation of JTAPI for private data. This implementation is targeted to independent switch vendors who want to use the private data programming mechanism to create private data packages, or application programmers who want to use or interpret private data that is provided in its raw form.

This document assumes a familiarity with the Java programming language and JTAPI and, for application programmers who want to use the TSAPI-specific information, TSAPI. For those readers interested in private data, it assumes a familiarity with TSAPI and its private data mechanism.

1 Hereafter, the phrase "Telephony Services implementation of JTAPI" indicates the Lucent Technologies PassageWay Telephony Services implementation of JTAPI.

Finding What You Need (Generic JTAPI and DEFINITY-Specific)

If you are an application programmer using the Telephony Services generic implementation of JTAPI to develop applications for all switches for which there is a PassageWay Telephony Services driver, or for the DEFINITY switch and the G3PD, find the appropriate description and refer to the associated chapters in the table below:

Intended Audience	This implementation provides:	Refer to Chapter(s):
Who: an application programmer **What:** Telephony Services generic implementation of JTAPI **To do:** programming with JTAPI for any switch for which there is a PassageWay Telephony Services driver **Be familiar with:** ■ Java ■ JTAPI ■ TSAPI (optional)	1) A programming environment for applications used with any switch for which there is a PassageWay Telephony Services driver; 2) A programming environment that makes available TSAPI-specific information (optional)	■ Chapter 1, "Telephony Services Implementation of JTAPI for All Switches and the DEFINITY Switch" ■ Chapter 2, "Using Telephony Services Extensions to JTAPI Exceptions" (optional)
Who: an application programmer **What:** Telephony Services implementation of JTAPI **To do:** programming with JTAPI for applications used with the DEFINITY switch and the G3PD **Be familiar with:** ■ Java ■ JTAPI ■ TSAPI (optional)	1) A programming environment for applications used with the DEFINITY switch and the G3PD; 2) A programming environment that makes available DEFINITY-specific features; 3) A programming environment that makes available TSAPI-specific information (optional)	■ Chapter 1, "Telephony Services Implementation of JTAPI for All Switches and the DEFINITY Switch" ■ Chapter 3, "Using Telephony Services DEFINITY-Specific Extensions to JTAPI" ■ Chapter 2, "Using Telephony Services Extensions to JTAPI Exceptions" (optional)

Finding What You Need (Non-DEFINITY Private Data)

If you are an independent switch vendor who is using the JTAPI private data programming environment to develop a private data package for non-DEFINITY switches, or an application programmer who is using or interpreting private data in a raw form, find the appropriate description and refer to the associated chapters in the table below:

Intended Audience	This implementation provides:	Refer to Chapter(s):
Who: An independent switch vendor **What:** Telephony Services implementation of JTAPI for applications using private data **To do:** Programming with JTAPI and TSAPI to produce private data packages for applications used with non-DEFINITY switches and their associated drivers **Be familiar with:** ■ Java ■ JTAPI ■ TSAPI ■ TSAPI's private data mechanism	A programming environment to produce private data packages that may be used with any non-DEFINITY switch and its associated PassageWay Telephony Services driver	■ Chapter 4, "Using Telephony Services Private Data Extensions to JTAPI" ■ Appendix A, "Telephony Services Implementation of JTAPI for Private Data"
Who: An application programmer **What:** Telephony Services implementation of JTAPI for applications using private data **To do:** Programming with JTAPI and TSAPI to use or interpret private data for applications used with non-DEFINITY switches and their associated drivers **Be familiar with:** ■ Java ■ JTAPI ■ TSAPI ■ TSAPI's private data mechanism	A programming environment to use or interpret private data, in a raw form, without an intermediate private data package, for applications used with any non-DEFINITY switch and its associated PassageWay Telephony Services driver	■ Chapter 4, "Using Telephony Services Private Data Extensions to JTAPI" ■ Appendix A, "Telephony Services Implementation of JTAPI for Private Data"

Navigating through the Document

This document is presented in PDF format with hypertext links and thumbnails for easy viewing and printing. Hypertext links are inserted so that you can easily navigate through the document by moving the hand symbol and clicking on the desired subject. You can also navigate through the document using the thumbnails of Adobe™ Acrobat™ Reader. After opening the PDF file with Adobe Reader, you can click on the second icon to see the list of items presented in outline format. If you click on one of the items in the list, you will be brought to the associated subject.

Related Documents

There is one other document related to JTAPI that is provided with the SDK.

Java Telephony API Programmer's Reference (JTAPI 1.2 Early Access)

This document consists of Sun Microsystem's Java Telephony API (JTAPI) specification files that are available to you from the Sun Microsystem Java Telephony API web site. This document is an early access version of the JTAPI1.2 specification. To obtain the very latest HTML files, go directly to the web site, **http://java.sun.com/products/jtapi**.

The following documents provide reference material about DEFINITY and Telephony Services Application Programming Interface (TSAPI) respectively.

PassageWay Telephony Services Solution DEFINITY ECS Programmer's Guide

This document presents information about DEFINITY switch administration and switch interactions.

Telephony Services Application Programming Interface (TSAPI) Version 2

This document presents information about how Telephony Services and TSAPI support telephony control capabilities in a generic, switch-independent way (i.e., support PBXs from various vendors). The architecture allows the incorporation of vendor-specific switch drivers to deliver Telephony Services across various switch environments.

Telephony Services Implementation of JTAPI

1

Contents

Telephony Services Implementation of JTAPI

1

Telephony Services Implementation of JTAPI for All Switches and the DEFINITY Switch

This chapter presents the level of support provided by the Telephony Services implementation of JTAPI for JTAPI interfaces and associated methods:

- for all switches for which there is a Telephony Services driver

  The information contained in the column entitled "**Supported for All Switches**" in the following tables represents the features supported by the Telephony Services generic implementation of JTAPI for any switch for which there is a PassageWay Telephony Services driver.

- for the DEFINITY switch and the G3PD driver

  The information contained in the column entitled "**Supported for the DEFINITY Switch**" in the following tables represents the features supported by the Telephony Services implementation of JTAPI for the DEFINITY switch and the G3PD driver.

This information is organized by JTAPI package, for example, Table 1-1 lists each interface and its associated method from the JTAPI Core package; Table 1-2 lists each interface and its associated method from the JTAPI CallCenter Capabilities package, and so forth.

Numerous tables are followed by implementation notes. If a number follows an interface (e.g., **JtapiPeer**[1]) or a method (e.g., getServices[2]), see the appropriate "Implementation Notes" section that follows the table to obtain additional information about the interfaces and/or their respective methods.

NOTE:
If you are an independent switch vendor who is using the JTAPI private data programming environment to develop a private data package for non-DEFINITY switches, or an application programmer who is using or interpreting private data in a raw form, ignore this chapter and refer to Chapter 4, "Telephony Services Private Data Extensions to JTAPI."

Support for JTAPI Core Package

The following table lists each JTAPI interface from the JTAPI Core Package, (e.g., **Address**), followed by its associated method(s), (e.g., getName, getProvide, getTerminals, and so forth), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

Table 1-1. Support for JTAPI Core Package

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
Address	√	√
getName	√	√
getProvider	√	√
getTerminals	√	√
getConnections	√	√
addObserver	√	√
getObservers	√	√
removeObserver	√	√
addCallObserver	√	√
getCallObservers	√	√
removeCallObserver	√	√
getAddressCapabilities	√	√
AddressObserver	√	√
addressChangedEvent	√	√
Call	√	√
getConnections	√	√
getProvider	√	√
getState	√	√
connect	√	√
addObserver	√	√
getObservers	√	√
removeObserver	√	√
getCallCapabilities	√	√
CallObserver	√	√
callChangedEvent	√	√
Connection	√	√

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
getState	√	√
getCall	√	√
getAddress	√	√
getTerminalConnections	√	√
disconnect	√	√
getConnectionCapabilities	√	√
JtapiPeer[1]	√	√
getName	√	√
getServices[2]	√	√
getProvider[3]	√	√
Provider	√	√
getState	√	√
getName	√	√
getCalls	√	√
getAddress	√	√
getAddresses	√	√
getTerminals	√	√
getTerminal	√	√
shutdown	√	√
createCall	√	√
addObserver	√	√
getObservers	√	√
removeObserver	√	√
getProviderCapabilities	√	√
getCallCapabilities	√	√
getConnectionCapabilities	√	√
getAddressCapabilities	√	√
getTerminalConnectionCapabilities	√	√
getTerminalCapabilities	√	√
ProviderObserver	√	√
providerChangedEvent	√	√
Terminal	√	√
getName	√	√
getProvider	√	√
getAddresses	√	√
getTerminalConnections	√	√
addObserver	√	√
getObservers	√	√
removeObserver	√	√
addCallObserver	√	√
getCallObservers	√	√
removeCallObserver	√	√
getTerminalCapabilities	√	√

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
TerminalConnection	√	√
getState	√	√
getTerminal	√	√
getConnection	√	√
answer	√	√
getTerminalConnectionCapabilities	√	√
TerminalObserver	√	√
terminalChangedEvent	√	√

Implementation Notes

1. Obtain a JtapiPeer object using the JtapiPeerFactory class. The TsapiPeer class represents this implementation of the JtapiPeer. To obtain TsapiPeer, do:

   ```
   JtapiPeerFactory.getJtapiPeer(com.lucent.jtapi.tsapi.TsapiPeer)
   ```

2. The **JtapiPeer**/getServices method returns an array of service names that can be used to build the String needed to be passed to JtapiPeer.getProvider(). These Strings are the Telephony Services server Tlink names.

3. The String provided by **JtapiPeer**/getProvider must contain a Telephony Services server Tlink name as well as a Windows NT login and password. The format of the String must be:

   ```
   <tlink>;login=<loginID>;passwd=<pw>
   ```

Support for JTAPI Call Center Package

The following table lists each JTAPI interface from the JTAPI Call Center Package, (e.g., **ACDAddress**), followed by its associated method(s), (e.g., getLoggedOnAgents, getNumberQueued, getOldestCallQueued, and so forth), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

Table 1-2. Support for JTAPI CallCenter Package

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
ACDAddress	√	√
getLoggedOnAgents[1]	√	√
getNumberQueued[2]	√	√
getOldestCallQueued		
getRelativeQueueLoad		
getQueueWaitTime		
getACDManagerAddress		
ACDAddressObserver	√	√
ACDConnection		√
getACDManagerConnection		√
ACDManagerAddress		√
getACDAddresses		
ACDManagerConnection		√
getACDConnections		√
AgentObject	√	√
setState	√	√
getState	√	√
getAgentID[3]	√	√
getACDAddress	√	√
getAgentAddress	√	√
getAgentTerminal	√	√
AgentTerminal	√	√
addAgent	√	√
removeAgent	√	√
getAgents	√	√
AgentTerminalObserver	√	√
CallCenterAddress	√	√
addCallObserver	√	√
CallCenterCall	√	√
connectPredictive[4]	√	√
setApplicationData		

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
getApplicationData		
getTrunks	√	√
CallCenterCallObserver	√	√
CallCenterProvider	√	√
getRouteableAddresses	√	√
getACDAddresses		√
getACDManagerAddresses		√
CallCenterTrunk	√	√
getName	√	√
getState	√	√
getType	√	√
getCall	√	√
RouteAddress	√	√
registerRouteCallback[5]	√	√
cancelRouteCallback	√	√
getRouteCallback	√	√
getActiveRouteSessions	√	√
RouteCallback	√	√
routeEvent	√	√
reRouteEvent	√	√
routeUsedEvent	√	√
routeEndEvent	√	√
routeCallbackEndedEvent	√	√
RouteSession	√	√
getRouteAddress	√	√
selectRoute[6]	√	√
endRoute	√	√
getState	√	√
getCause	√	√

Implementation Notes

1 The **ACDAddress**/getLoggedOnAgents method is fully supported for the DEFINITY switch. For other switches, it returns the sum of (a) those agents that were logged in through the application and (b) those agents that were logged in after an ACDAddressObserver was added to the application.

2 The **ACDAddress**/getNumberQueued method is fully supported for the DEFINITY switch. For other switches, it returns the number of calls queued reported in the last queued event. This may not be accurate since some of the calls may have been subsequently dequeued.

Implementation Notes (Continued)

3 The **AgentObject**/getAgentID method returns a null String for the DEFINITY switch.

4 The **CallCenterCall**/connectPredictive method is supported for the DEFINITY switch and for other switches; however, the answeringEndpointType parameter is not supported. For the DEFINITY switch, the maxRings and answeringTreatment parameters are supported. For other switches, the maxRings and answeringTreatment parameters are not supported.

For the DEFINITY switch, if the Call is observed and the ACDAddress or AgentTerminal is also call observed, then two unique Call objects will be created that are associated with the same real call.

Other methods must be used to determine that there are two Call objects representing the same real call:

- One way to do this, if the called address is unique among all calls, is to use the Call.getCalledAddress() method.
- Another way is to use the UserToUserInfo DEFINITY-specific extension. The application can send a unique ID in the UserToUserInfo with the connectPredictive and this ID will be reported in call events for the ACDAddress or AgentTerminal. The UserToUserInfo can also be retrieved directly from the Calls.

In any case, both Call objects and all Connections and TerminalConnections in both Calls are valid. Valid requests may be made of any of the objects.

5 The **RouteAddress**/registerRouteCallback method is supported for the DEFINITY switch and other switches; however, only one RouteCallback may be registered for an Address at a time.

6 The **RouteSession**/selectRoute method is supported for the DEFINITY switch and other switches; however, only the first route specified in the routeSelected parameter is used. The subsequent routes are ignored.

Support for JTAPI Call Center Capabilities Package

The following table lists each JTAPI interface from the JTAPI Call Center Capabilities Package, (e.g., **ACDAddressCapabilities**), followed by its associated method(s), (e.g., canGetLoggedOnAgents, canGetNumberQueued, and so forth), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

Table 1-3. Support for JTAPI CallCenter Capabilities Package

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
ACDAddressCapabilities	√	√
canGetLoggedOnAgents	√	√
canGetNumberQueued	√	√
canGetOldestCallQueued	√	√
canGetRelativeQueueLoad	√	√
canGetQueueWaitTime	√	√
canGetACDManagerAddress	√	√
ACDConnectionCapabilities	√	√
canGetACDManagerConnection	√	√
ACDManagerAddressCapabilities	√	√
canGetACDAddresses	√	√
ACDManagerConnectionCapabilities	√	√
canGetACDConnections	√	√
AgentTerminalCapabilities	√	√
canHandleAgents	√	√
CallCenterAddressCapabilities	√	√
canAddCallObserver	√	√
CallCenterCallCapabilities	√	√
canConnectPredictive	√	√
canHandleApplicationData	√	√
canGetTrunks	√	√
CallCenterProviderCapabilities	√	√
canGetRouteableAddresses	√	√
canGetACDAddresses	√	√
canGetACDManagerAddresses	√	√
canGetTrunks	√	√
RouteAddressCapabilities	√	√
canRouteCalls	√	√

Support for JTAPI Call Center Events Package

The following table lists each JTAPI interface from the JTAPI Call Center Events Package, (e.g., **ACDAddrBusyEv**, **ACDAddrLoggedOffEv**, **ACDAddrLoggedOnEv**, and so forth), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

NOTE:
If a JTAPI Call Center Event is supported, all associated methods are also supported.

Table 1-4. Support for JTAPI CallCenter Events Package

JTAPI Interfaces	Supported for All Switches	Supported for the DEFINITY Switch
ACDAddrBusyEv		√
ACDAddrLoggedOffEv	√	√
ACDAddrLoggedOnEv	√	√
ACDAddrNotReadyEv[1]	√	√
ACDAddrReadyEv[1]	√	√
ACDAddrUnknownEv	√	√
ACDAddrWorkNotReadyEv[1]	√	√
ACDAddrWorkReadyEv[1]	√	√
AgentTermBusyEv		√
AgentTermLoggedOffEv	√	√
AgentTermLoggedOnEv	√	√
AgentTermNotReadyEv[1]	√	√
AgentTermReadyEv[1]	√	√
AgentTermUnknownEv	√	√
AgentTermWorkNotReadyEv[1]	√	√
AgentTermWorkReadyEv[1]	√	√
CallCentCallAppDataEv		
RouteCallbackEndedEvent	√	√
RouteEndEvent	√	√
RouteEvent	√	√
RouteSessionEvent	√	√
RouteUsedEvent	√	√

Implementation Notes

1 These events are not supported for the DEFINITY switch. They will be generated by the implementation and sent to the application when an explicit state change is requested by the application.

Support for JTAPI Call Control Package

The following table lists each JTAPI interface from the JTAPI Call Control Package, (e.g., **CallControlAddress**), followed by its associated method(s), (e.g., setForwarding, getForwarding, cancelForwarding, and so forth), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

Table 1-5. Support for JTAPI CallControl Package

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
CallControlAddress	√	√
setForwarding[1]	√	√
getForwarding	√	√
cancelForwarding	√	√
getDoNotDisturb[2]	√	√
setDoNotDisturb[2]	√	√
getMessageWaiting	√	√
setMessageWaiting	√	√
CallControlAddressObserver	√	√
CallControlCall	√	√
getCallingAddress	√	√
getCallingTerminal	√	√
getCalledAddress	√	√
getLastRedirectedAddress	√	√
addParty		
drop	√	√
offHook		
conference	√	√
transfer(Call otherCall)	√	√
transfer(String address)		
setConferenceController	√	√
getConferenceController	√	√
setTransferController	√	√
getTransferController	√	√
setConferenceEnable	√	√
getConferenceEnable	√	√
setTransferEnable	√	√
getTransferEnable	√	√
consult (TerminalConnection termconn, String address)	√	√
consult (TerminalConnection termconn)		
CallControlCallObserver	√	√
CallControlConnection	√	√
getCallControlState	√	√

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
accept		
reject		
redirect	√	√
addToAddress		
park		
CallControlTerminal	√	√
getDoNotDisturb[2]	√	√
setDoNotDisturb[2]	√	√
pickup (Connection pickConnection, Address terminalAddress)	√	√
pickup (TerminalConnection pickTermConn, Address terminalAddress)	√	√
pickup (Address pickAddress, Address terminalAddress)	√	√
pickupFromGroup(String pickupGroup, Address terminalAddress)		
pickupFromGroup(Address terminalAddress)	√	√
CallControlTerminalConnection	√	√
getCallControlState	√	√
hold	√	√
unhold	√	√
join		√
leave		√
CallControlTerminalObserver	√	√

Implementation Notes

1 The **CallControlAddress**/setForwarding method is fully supported for the DEFINITY switch and for other switches; however, the SPECIFIC_ADDRESS filter instruction is not supported. The ALL_CALLS filter instruction is only supported when the filter type is FORWARD_UNCONDITIONALLY. The DEFINITY switch only supports the ALL_CALLS filter instruction with the FORWARD_UNCONDITIONALLY filter type.

Implementation Notes (Continued)

2 The following methods are paired synonyms:

CallControlAddress/getDoNotDisturb
CallControlTerminal/getDoNotDisturb

CallControlAddress/setDoNotDisturb
CallControlTerminal/setDoNotDisturb

For these methods, there is no distinction between an Address and a Terminal. CallControlAddress.getDoNotDisturb() and CallControlTerminal.getDoNotDisturb() always return equivalent values.

Support for JTAPI Call Control Capabilities Package

The following table lists each JTAPI interface from the JTAPI Call Control Capabilities Package, (e.g., **CallControlAddressCapabilities**), followed by its associated method(s), (e.g., canSetForwarding, canGetForwarding, canCancelForwarding, and so forth), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

Table 1-6. Support for JTAPI CallControl Capabilities Package

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
CallControlAddressCapabilities	√	√
canSetForwarding	√	√
canGetForwarding	√	√
canCancelForwarding	√	√
canGetDoNotDisturb	√	√
canSetDoNotDisturb	√	√
canGetMessageWaiting	√	√
canSetMessageWaiting	√	√
CallControlCallCapabilities	√	√
canDrop	√	√
canOffHook	√	√
canSetConferenceController	√	√
canSetTransferController	√	√
canSetTransferEnable	√	√
canSetConferenceEnable	√	√
canTransfer	√	√
canConference	√	√
canAddParty	√	√
canConsult	√	√
CallControlConnectionCapabilities	√	√
canRedirect	√	√
canAddToAddress	√	√
canAccept	√	√
canReject	√	√
canPark	√	√
CallControlTerminalCapabilities	√	√
canGetDoNotDisturb	√	√
canSetDoNotDisturb	√	√
canPickup	√	√
canPickupFromGroup	√	√
CallControlTerminalConnectionCapabilities	√	√

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
canHold	√	√
canUnhold	√	√
canJoin	√	√
canLeave	√	√

Support for JTAPI Call Control Events Package

The following table lists each JTAPI interface from the JTAPI Call Control Events Package, (e.g., **CallCtlAddrDoNotDisturbEv**, **CallCtlAddrForwardEv**, **CallCtlAddrMessageWaitingEv**, and so forth), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

NOTE:
If a JTAPI Call Control Event is supported, all associated methods are also supported.

Table 1-7. Support for JTAPI CallControl Events Package

JTAPI Interfaces	Supported for All Switches	Supported for the DEFINITY Switch
CallCtlAddrDoNotDisturbEv[1]	√	√
CallCtlAddrForwardEv	√	√
CallCtlAddrMessageWaitingEv	√	√
CallCtlConnAlertingEv	√	√
CallCtlConnDialingEv		
CallCtlConnDisconnectedEv	√	√
CallCtlConnEstablishedEv	√	√
CallCtlConnFailedEv	√	√
CallCtlConnInitiatedEv	√	√
CallCtlConnNetworkAlertingEv	√	√
CallCtlConnNetworkReachedEv	√	√
CallCtlConnOfferedEv		
CallCtlConnQueuedEv	√	√
CallCtlConnUnknownEv	√	√
CallCtlTermConnBridgedEv	√	√
CallCtlTermConnDroppedEv	√	√
CallCtlTermConnHeldEv	√	√
CallCtlTermConnInUseEv		
CallCtlTermConnRingingEv	√	√
CallCtlTermConnTalkingEv	√	√
CallCtlTermConnUnknownEv	√	√

Implementation Notes

1 The **CallCtrlAddrDoNotDisturbEv** event is sent even if DoNotDisturb was changed using CallControlTerminal.setDoNotDisturb(). For DoNotDisturb, there is no distinction between an Address and a Terminal.

Support for JTAPI Capabilities Package

The following table lists each JTAPI interface from the JTAPI Capabilities Package, (e.g., **AddressCapabilities**), followed by its associated method(s), (e.g., isObservable), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

Table 1-8. Support for JTAPI Capabilities Package

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
AddressCapabilities	√	√
isObservable	√	√
CallCapabilities	√	√
canConnect	√	√
isObservable	√	√
ConnectionCapabilities	√	√
canDisconnect	√	√
ProviderCapabilities	√	√
isObservable	√	√
TerminalCapabilities	√	√
isObservable	√	√
TerminalConnectionCapabilities	√	√
isObservable	√	√

Support for JTAPI Events Package

The following table lists each JTAPI interface from the JTAPI Events Package, (e.g., **AddObservationEndedEv**, **CallActiveEv**, **CallInvalid**, and so forth), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

NOTE:
If a JTAPI Event is supported, all associated methods are also supported.

Table 1-9. Support for JTAPI Events Package

JTAPI Interfaces	Supported for All Switches	Supported for the DEFINITY Switch
AddrObservationEndedEv	√	√
CallActiveEv	√	√
CallInvalidEv	√	√
CallObservationEndedEv	√	√
ConnAlertingEv	√	√
ConnConnectedEv	√	√
ConnCreatedEv	√	√
ConnDisconnectedEv	√	√
ConnFailedEv	√	√
ConnInProgressEv	√	√
ConnUnknownEv	√	√
ProvInServiceEv	√	√
ProvObservationEndedEv	√	√
ProvOutOfServiceEv	√	√
ProvShutdownEv	√	√
TermConnActiveEv	√	√
TermConnCreatedEv	√	√
TermConnDroppedEv	√	√
TermConnPassiveEv	√	√
TermConnRingingEv	√	√
TermConnUnknownEv	√	√
TermObservationEndedEv	√	√

Support for JTAPI Media Package

The following table lists each JTAPI interface from the JTAPI Media Package, (e.g., **MediaCallObserver**), followed by its associated method(s), (if any), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

Table 1-10. Support for JTAPI Media Package

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
MediaCallObserver		√
MediaTerminalConnection		√
getMediaAvailability		
getMediaState		
useDefaultSpeaker		
useRecordURL		
useDefaultMicrophone		
usePlayURL		
startPlaying		
stopPlaying		
startRecording		
setDtmfDetection		
generateDtmf		√

Support for JTAPI Media Capabilities Package

The following table lists each JTAPI interface from the JTAPI Media Capabilities Package, (e.g., **MediaTerminalConnectionCapabilities**), followed by its associated method(s), (e.g., canUseDefaultSpeaker, canUseDefaultMicrophone, canUseRecordURL, and so forth), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

Table 1-11. Support for JTAPI Media Capabilities Package

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
MediaTerminalConnectionCapabilities	√	√
canUseDefaultSpeaker	√	√
canUseDefaultMicrophone	√	√
canUseRecordURL	√	√
canUsePlayURL	√	√
canStartPlaying	√	√
canStopPlaying	√	√
canStartRecording	√	√
canStopRecording	√	√
canDetectDtmf	√	√
canGenerateDtmf	√	√

Support for JTAPI Media Events Package

The following table lists each JTAPI interface from the JTAPI Media Events Package, (e.g., **MediaTermConnAvailable**, **MediaTermConnDtmfEv**, **MediaTermConnEv**, and so forth), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

NOTE:
If a JTAPI Media Event is supported, all associated methods are also supported.

Table 1-12. Support for JTAPI Media Events Package

JTAPI Interfaces	Supported for All Switches	Supported for the DEFINITY Switch
MediaTermConnAvailableEv		
MediaTermConnDtmfEv[1]		√
MediaTermConnStateEv		
MediaTermConnUnavailableEv		

Implementation Notes

1 Although the **MediaTermConnDtmfEv** interface has been defined as a TerminalConnection event, the TerminalConnection field will be null. The Call field will be filled in with the call to which the DTMF digits have been applied.

This event is sent only when a DTMF detector is attached to the call and DTMF tones are detected. The tone detector is disconnected when the far end answers or "#" is detected. This event is used in conjunction with the DEFINITY-specific extension **LucentRouteSession**/selectRouteAndCollect.

Support for JTAPI Phone Package

NOTE:
The JTAPI Phone Package interfaces and methods are not supported.

Support for JTAPI Phone Capabilities Package

NOTE:
The JTAPI Phone Package interfaces and methods are not supported.

Support for JTAPI Phone Events Package

NOTE:
The JTAPI Phone Events Package interfaces and methods are not supported.

Support for JTAPI Private Data Package

The following table lists each JTAPI interface from the JTAPI Private Data Package, (e.g., **PrivateData**), followed by its associated method(s), (e.g., getPrivateData, and so forth), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

Table 1-13. Support for JTAPI Private Data Package

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
PrivateData	√	
getPrivateData	√	
setPrivateData[1]	√	
sendPrivateData[1]	√	

Implementation Notes

1 For the **PrivateData**/setPrivateData and **PrivateData**/sendPrivateData methods, the private data Object parameter must be an instance of TsapiPrivate.

Support for JTAPI Private Data Capabilities Package

The following table lists each JTAPI interface from the JTAPI Private Data Capabilities Package, (e.g., **PrivateDataCapabilities**), followed by its associated method(s), (e.g., canSetPrivateData, and so forth), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

Table 1-14. Support for JTAPI Private Data Capabilities Package

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
PrivateDataCapabilities	√	√
canSetPrivateData	√	√
canGetPrivateData	√	√
canSendPrivateData	√	√

Support for JTAPI Private Data Events Package

The following table lists each JTAPI interface from the JTAPI Private Data Events Package, (e.g., **PrivateAddrEv**, **PrivateCallEv**, and so forth), and whether the implementation is supported for all switches and/or for the DEFINITY switch.

NOTE:
If a JTAPI Private Data Event is supported, all associated methods are also supported.

Table 1-15. Support for JTAPI Private Data Events Package

JTAPI Interfaces	Supported for All Switches	Supported for the DEFINITY Switch
PrivateAddrEv	√	
PrivateCallEv	√	
PrivateProvEv	√	
PrivateTermEv	√	

Using Telephony Services Extensions to JTAPI Exceptions

2

Contents

Using Telephony Services Extensions to JTAPI Exceptions

2

Using Telephony Services Extensions to JTAPI Exceptions

NOTE:
This chapter describes non-standard additions to JTAPI. This package is available only from the PassageWay Telephony Services implementation of JTAPI and is not available from any other implementation of JTAPI.

This chapter is optional. It contains the Telephony Services extensions to JTAPI exceptions that can be used to program applications for any switch for which there is a PassageWay Telephony Services driver.

Who Should Be Using These Extensions?

An application programmer who, in addition to using the standard JTAPI package, wants additional TSAPI-specific information about the JTAPI exceptions to develop applications which will be used with any switch for which there is a PassageWay Telephony Services driver. It is assumed that this individual has a familiarity with the Java programming language, JTAPI, and TSAPI.

NOTE:
If you are an application programmer who is using JTAPI to develop applications for the DEFINITY switch, also refer to Chapter 1, "Telephony Services Implementation of JTAPI for All Switches and the DEFINITY Switch." If you want to take advantage of DEFINITY-specific features that are not accessible through standard JTAPI, refer to Chapter 3, "Telephony Services DEFINITY-Specific Extensions to JTAPI."

NOTE:
If you are an independent switch vendor who is using the JTAPI private data programming environment to develop a private data package for non-DEFINITY switches, or an application programmer who is using or interpreting private data in a raw form, ignore this chapter and refer to Chapter 4, "Telephony Services Private Data Extensions to JTAPI."

What are the Extensions?

Telephony Services extensions to the JTAPI exceptions provide more detailed error information than is defined in JTAPI. These extensions consist of the CSTA and ACS error codes provided by TSAPI.

NOTE:
For information about Computer-Supported Telecommunications Applications (CSTA) and API Control Services (ACS) error codes, refer to the *Telephony Services Application Programming Interface (TSAPI) Version 2*.

package com.lucent.jtapi.tsapi

Interface Index

- ITsapiException

Exception Index

- TsapiInvalidArgumentException
- TsapiInvalidObjectException
- TsapiInvalidPartyException
- TsapiInvalidStateException
- TsapiMethodNotSupportedException
- TsapiPlatformException
- TsapiPrivilegeViolationException
- TsapiProviderUnavailableException
- TsapiResourceUnavailableException

Interface com.lucent.jtapi.tsapi.ITsapiException

public interface **ITsapiException**

The ITsapiException interface adds an errorType and errorCode to all Jtapi exceptions. When the errorType is ACS or CSTA, the errorCode will contain the Tsapi ACS or CSTA error code which is documented in the Passageway Telephony Services Network Manager's Guide in the TroubleShooting section (netmangd.pdf).

Variable Index

Method Index

Variables

o **NORMAL**

```
public static final int NORMAL
```

Error Type of Normal.

- **ACS**

```
public static final int ACS
```

Error Type of ACS.

- **CSTA**

```
public static final int CSTA
```

Error Type of CSTA.

- **JTAPI**

```
public static final int JTAPI
```

Failed to meet some Jtapi condition.

- **INTERNAL**

```
public static final int INTERNAL
```

Failure is internal to this Jtapi implementation.

- **EC_NORMAL**

```
public static final int EC_NORMAL
```

Error Code of NORMAL.

- **EC_INVALID_CONF**

```
public static final int EC_INVALID_CONF
```

Error Code implying confirmation is invalid.

- **EC_PROVIDER_OUT_OF_SERVICE**

```
public static final int EC_PROVIDER_OUT_OF_SERVICE
```

Error Code implying Provider is OUT_OF_SERVICE.

Methods

o **getErrorType**

```
public abstract int getErrorType()
```

Returns the error type.

o **getErrorCode**

```
public abstract int getErrorCode()
```

Returns the error code.

Class com.lucent.jtapi.tsapi.TsapiInvalidArgumentException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----javax.telephony.InvalidArgumentException
                           |
                           +----com.lucent.jtapi.tsapi.TsapiInvalidArgumentException
```

---

public final class **TsapiInvalidArgumentException**
extends InvalidArgumentException
implements ITsapiException

TsapiInvalidArgumentException extends Jtapi InvalidArgumentException to add implementation specific errorType and errorCode.

See ITsapiException for details on errorType and errorCode.

Method Index

Methods

o **getErrorType**

```
public int getErrorType()
```

Returns the error type.

o **getErrorCode**

```
public int getErrorCode()
```

Returns the error code.

Class com.lucent.jtapi.tsapi.TsapiInvalidPartyException

```
java.lang.Object
   |
   +----java.lang.Throwable
            |
            +----java.lang.Exception
                     |
                     +----javax.telephony.InvalidPartyException
                              |
                              +----com.lucent.jtapi.tsapi.TsapiInvalidPartyException
```

---

public final class **TsapiInvalidPartyException**
extends InvalidPartyException
implements ITsapiException

TsapiInvalidPartyException extends Jtapi InvalidPartyException to add implementation specific errorType and errorCode.

See ITsapiException for details on errorType and errorCode.

Method Index

Methods

o **getErrorType**

```
public int getErrorType()
```

Returns the error type.

o **getErrorCode**

```
public int getErrorCode()
```

Returns the error code.

Class com.lucent.jtapi.tsapi.TsapiInvalidStateException

```
java.lang.Object
   |
   +----java.lang.Throwable
            |
            +----java.lang.Exception
                     |
                     +----javax.telephony.InvalidStateException
                              |
                              +----com.lucent.jtapi.tsapi.TsapiInvalidStateException
```

---

public final class **TsapiInvalidStateException**
extends InvalidStateException
implements ITsapiException

TsapiInvalidStateException extends Jtapi InvalidStateException to add implementation specific errorType and errorCode.

See ITsapiException for details on errorType and errorCode.

Method Index

Methods

o getErrorType

```
public int getErrorType()
```

Returns the error type.

o getErrorCode

```
public int getErrorCode()
```

Returns the error code.

Class com.lucent.jtapi.tsapi.TsapiMethodNotSupportedException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----javax.telephony.MethodNotSupportedException
                           |
                           +----com.lucent.jtapi.tsapi.TsapiMethodNotSupportedException
```

---

public final class **TsapiMethodNotSupportedException**
extends MethodNotSupportedException
implements ITsapiException

TsapiMethodNotSupportedException extends MethodNotSupportedException to add implementation specific errorType and errorCode.

See ITsapiException for details on errorType and errorCode.

Method Index

Methods

o **getErrorType**

```
public int getErrorType()
```

Returns the error type.

o **getErrorCode**

```
public int getErrorCode()
```

Returns the error code.

Class com.lucent.jtapi.tsapi.TsapiPlatformException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----javax.telephony.PlatformException
                                   |
                                   +----com.lucent.jtapi.tsapi.TsapiPlatformException
```

---

public final class **TsapiPlatformException**
extends PlatformException
implements ITsapiException

TsapiPlatformException extends Jtapi PlatformException to add implementation specific errorType and errorCode.

See ITsapiException for details on errorType and errorCode.

Method Index

Methods

o **getErrorType**

```
public int getErrorType()
```

Returns the error type.

o **getErrorCode**

```
public int getErrorCode()
```

Returns the error code.

Class com.lucent.jtapi.tsapi.TsapiPrivilegeViolationException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----javax.telephony.PrivilegeViolationException
                           |
                           +----com.lucent.jtapi.tsapi.TsapiPrivilegeViolationException
```

---

public final class **TsapiPrivilegeViolationException**
extends PrivilegeViolationException
implements ITsapiException

TsapiPrivilegeViolationException extends PrivilegeViolationException to add acs / csta error codes.

Method Index

Methods

o getErrorType

```
public int getErrorType()
```

Returns the error type.

o getErrorCode

```
public int getErrorCode()
```

Returns the error code.

Class com.lucent.jtapi.tsapi.TsapiProviderUnavailableException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----javax.telephony.ProviderUnavailableException
                                   |
                                   +----com.lucent.jtapi.tsapi.TsapiProviderUnavailableException
```

---

public final class **TsapiProviderUnavailableException**
extends ProviderUnavailableException
implements ITsapiException

TsapiProviderUnavailableException extends Jtapi ProviderUnavailableException to add implementation specific errorType and errorCode.

See ITsapiException for details on errorType and errorCode.

Method Index

Methods

o **getErrorType**

```
public int getErrorType()
```

Returns the error type.

- **getErrorCode**

```
public int getErrorCode()
```

Returns the error code.

Class com.lucent.jtapi.tsapi.TsapiResourceUnavailableException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----javax.telephony.ResourceUnavailableException
                           |
                           +----com.lucent.jtapi.tsapi.TsapiResourceUnavailableException
```

---

public final class **TsapiResourceUnavailableException**
extends ResourceUnavailableException
implements ITsapiException

TsapiResourceUnavailableException extends Jtapi ResourceUnavailableException to add implementation specific errorType and errorCode.

See ITsapiException for details on errorType and errorCode.

Method Index

Methods

o **getErrorType**

```
public int getErrorType()
```

Returns the error type.

o **getErrorCode**

```
public int getErrorCode()
```

Returns the error code.

Using Telephony Services DEFINITY-Specific Extensions to JTAPI

3

Contents

Using Telephony Services DEFINITY-Specific Extensions to JTAPI

3

Using Telephony Services DEFINITY-Specific Extensions

NOTE:
This chapter describes non-standard additions to JTAPI. This package is available only from the PassageWay Telephony Services implementation of JTAPI and is not available from any other implementation of JTAPI.

This chapter is optional. It is an intermediate private data package that allows programmers to access private data via Java interfaces rather than through raw private data bytes. It contains the DEFINITY-specific feature extensions that support the Telephony Services implementation of JTAPI intended for applications that operate solely with the DEFINITY switch.

Who Should Be Using These Extensions?

An application programmer using JTAPI to develop applications that will be used with the DEFINITY switch and the associated PassageWay Telephony Services driver (i.e., the G3PD). In addition, these applications will take advantage of DEFINITY-specific features that are not accessible through standard JTAPI. It is assumed that this individual has a familiarity with both the Java programming language and JTAPI.

NOTE:
If you are an application programmer who is using JTAPI to develop applications for any switch for which there is a PassageWay Telephony Services driver, ignore this chapter and refer to Chapter 1, “Telephony Services Implementation of JTAPI for All Switches and the DEFINITY Switch.” If you want additional TSAPI-specific information about the JTAPI exceptions that are not accessible through standard JTAPI, refer to Chapter 2, “ Using Telephony Services Extensions to JTAPI Exceptions.”

NOTE:
If you are an independent switch vendor who is using the JTAPI private data programming environment to develop a private data package for non-DEFINITY switches, or an application programmer who is using or interpreting private data in a raw form, ignore this chapter and refer to Chapter 4, "Telephony Services Private Data Extensions to JTAPI."

How Should the Extensions be Used?

The DEFINITY-specific extensions to JTAPI make available DEFINITY features beyond those provided by the standard Telephony Services implementation of JTAPI. The following table lists each DEFINITY feature that is available as an extension to JTAPI, its description, its associated class or interface, and the methods returned or used by methods in each appropriate class or interface.

Feature Name and Description	Class or Interface	Returned/Used by Methods in Class or Interface
Agent Work Mode Specifies the overriding mode of the Agent; affects the cycle of the possibly occurring Agents states		LucentAgent LucentAgentStateInfo LucentTerminal
Call Classifier Information Provides information on call classifier port usage	CallClassifierInfo	LucentProvider
Collect Digits Allows a route request to wait for a specified number of digits to be collected		LucentRouteSession
Dial-Ahead Digits Allows a route request to place digits in a dial-ahead buffer		LucentRouteSession
Direct Agent Calls Allows calls to be made to and from specific logged-in ACD Agents		LucentCall LucentRouteSession
Dropping Resources Allows specific switch resources to be dropped from the call		LucentConnection LucentTerminalConnection
Integrated Directory Name Allows the G3 Integrated Directory Database name to be returned		LucentAddress LucentTerminal
Look-Ahead Interflow Information May be used by a routing server application to determine the proper destination of a call	LookaheadInfo	LucentCallInfo OriginalCallInfo

Feature Name and Description	Class or Interface	Returned/Used by Methods in Class or Interface
Lucent Call Information Provides Lucent-specific call information on Call and CallControlCall events; information includes delivering ACD, distributing Address, originating Trunk, reason for last Call event, and other information	LucentCallInfo	Cast LucentCall and CallControlCall events to the LucentCallInfo interface for access
Message Waiting Application Information Indicates which types of applications have enabled message waiting		LucentAddress LucentAddressMsgWaitingEvent
Network Progress Information Contains supplementary call progress information from the ISDN Progress Indicator Information Element	NetworkProgressInfo	LucentConnNetworkReachedEvent
Original Call Information Contains information about the original call in conjunction with the Call.consult() service	OriginalCallInfo	LucentCallInfo
Priority Calls Enables priority calling		LucentCall LucentRouteSession
Supervisor Assist Calls Allows logged-in ACD Agents to place calls to a supervisor's extension		LucentCall
Switch Date and Time Information Returns the current date and time from the switch		LucentProvider
Trunk Group Information Provides information on trunk group usage	TrunkGroupInfo	LucentProvider
User Entered Code The code/digits that may have been entered by the caller through the G3 Call Prompting feature of the Collected Digits feature	UserEnteredCode	LucentCallInfo OriginalCallInfo
User-to-User Information An ISDN feature that allows end-to-end transmission of application data during call setup/teardown	UserToUserInfo	LucentCall LucentCallInfo LucentConnection LucentRouteSession LucentTerminalConnection OriginalCallInfo

package com.lucent.jtapi.tsapi

Interface Index

Class Index

Interface com.lucent.jtapi.tsapi.ITsapiAddress

public interface **ITsapiAddress**
extends Address, CallControlAddress, CallCenterAddress, RouteAddress

ITsapiAddress extends Jtapi Address, CallControlAddress, CallCenterAddress, RouteAddress.

This interface was added so that LucentAddress could extend it and migration of methods from LucentAddress to ITsapiAddress would not affect applications using LucentAddress. Methods in LucentAddress currently map to Tsapi Private Data for Definity. It is expected that once the functionality is part of Tsapi the methods will migrate.

Interface com.lucent.jtapi.tsapi.ITsapiAddressMsgWaitingEvent

public interface **ITsapiAddressMsgWaitingEvent**
extends CallCtlAddrMessageWaitingEv

ITsapiAddressMsgWaitingEvent implements Jtapi CallCtlAddrMessageWaitingEv.

This interface was added so that LucentAddressMsgWaitingEvent could extend it and migration of methods from LucentAddressMsgWaitingEvent to ITsapiAddressMsgWaitingEvent would not affect applications using LucentAddressMsgWaitingEvent. Methods in LucentAddressMsgWaitingEvent currently map to Tsapi Private Data for Definity. It is expected that once the functionality is part of Tsapi the methods will migrate.

Interface com.lucent.jtapi.tsapi.ITsapiAgent

public interface **ITsapiAgent**
extends Agent

ITsapiAgent extends Agent.

This interface was added so that LucentAgent could extend it and migration of methods from LucentAgent to ITsapiAgent would not affect applications using LucentAgent. Methods in LucentAgent currently map to Tsapi Private Data for Definity. It is expected that once the functionality is part of Tsapi the methods will migrate.

Interface com.lucent.jtapi.tsapi.ITsapiCall

public interface **ITsapiCall**
extends ITsapiCallInfo, Call, CallControlCall, CallCenterCall

ITsapiCall extends Jtapi Call, CallControlCall, CallCenterCall.

This interface was added so that LucentCall could extend it and migration of methods from LucentCall to ITsapiCall would not affect applications using LucentCall. Methods in LucentCall currently map to Tsapi Private Data for Definity. It is expected that once the functionality is part of Tsapi the methods will migrate.

Interface com.lucent.jtapi.tsapi.ITsapiCallInfo

public interface **ITsapiCallInfo**

ITsapiCallInfo adds new call information for Call and events

This interface was added so that LucentCallInfo could extend it and migration of methods from LucentCallInfo to ITsapiCallInfo would not affect applications using LucentCallInfo. Methods in LucentCallInfo currently map to Tsapi Private Data for Definity. It is expected that once the functionality is part of Tsapi the methods will migrate.

Interface com.lucent.jtapi.tsapi.ITsapiConnNetworkReachedEvent

public interface **ITsapiConnNetworkReachedEvent**
extends CallCtlConnNetworkReachedEv

ITsapiConnNetworkReachedEvent extends Jtapi CallCtlConnNetworkReachedEv.

This interface was added so that LucentConnNetworkReachedEvent could extend it and migration of methods from LucentConnNetworkReachedEvent to ITsapiConnNetworkReachedEvent would not affect applications using LucentConnNetworkReachedEvent.Methods in LucentConnNetworkReachedEvent currently map to Tsapi Private Data for Definity. It is expected that once the functionality is part of Tsapi the methods will migrate.

Interface com.lucent.jtapi.tsapi.ITsapiConnection

public interface **ITsapiConnection**
extends Connection, CallControlConnection

ITsapiConnection extends Jtapi Connection and CallControlConnection.

This interface was added so that LucentConnection could extend it and migration of methods from LucentConnection to ITsapiConnection would not affect applications using LucentConnection. Methods in LucentConnection currently map to Tsapi Private Data for Definity. It is expected that once the functionality is part of Tsapi the methods will migrate.

Interface com.lucent.jtapi.tsapi.ITsapiRouteSession

public interface **ITsapiRouteSession**
extends RouteSession, ITsapiCallInfo

ITsapiRouteSession extends Jtapi RouteSession.

This interface was added so that LucentRouteSession could extend it and migration of methods from LucentRouteSession to ITsapiRouteSession would not affect applications using LucentRouteSession.Methods in LucentRouteSession currently map to Tsapi Private Data for Definity. It is expected that once the functionality is part of Tsapi the methods will migrate.

Interface com.lucent.jtapi.tsapi.ITsapiTerminal

public interface **ITsapiTerminal**
extends Terminal, CallControlTerminal, AgentTerminal

ITsapiTerminal extends Terminal, CallControlTerminal and AgentTerminal.

This interface was added so that LucentTerminal could extend it and migration of methods from LucentTerminal to ITsapiTerminal would not affect applications using LucentTerminal. Methods in LucentTerminal currently map to Tsapi Private Data for Definity. It is expected that once the functionality is part of Tsapi the methods will migrate.

Interface com.lucent.jtapi.tsapi.ITsapiTerminalConnection

public interface **ITsapiTerminalConnection**
extends TerminalConnection, CallControlTerminalConnection,
MediaTerminalConnection

ITsapiTerminalConnection extends TerminalConnection,
CallControlTerminalConnection, and MediaTerminalConnection.

This interface was added so that LucentTerminalConnection could extend it and migration of methods from LucentTerminalConnection to ITsapiTerminalConnection would not affect applications using LucentTerminalConnection. Methods in LucentTerminalConnection currently map to Tsapi Private Data for Definity. It is expected that once the functionality is part of Tsapi the methods will migrate.

Interface com.lucent.jtapi.tsapi.LucentAddress

public interface **LucentAddress**
extends ITsapiAddress

This interface add Lucent−specific methods to the Address interface.

Variable Index

Method Index

Variables

o **MWI_MCS**

```
public static final int MWI_MCS
```

The message waiting indicator has been enabled via Message Center.

o **MWI_VOICE**

```
public static final int MWI_VOICE
```

The message waiting indicator has been enabled via Voice Messaging.

- **MWI_PROPMGT**

```
public static final int MWI_PROPMGT
```

The message waiting indicator has been enabled via Property Management.

- **MWI_LWC**

```
public static final int MWI_LWC
```

The message waiting indicator has been enabled via Leave Word Calling.

- **MWI_CTI**

```
public static final int MWI_CTI
```

The message waiting indicator has been enabled via CTI.

Methods

- **getMessageWaitingBits**

```
public abstract int getMessageWaitingBits() throws TsapiMethodNotSupportedException
```

Returns a bit−mask indicating which applications have enabled the message waiting indicator at this Address. Its value is a logical−OR combination of MWI_MCS, MWI_VOICE, MWI_PROPMGT, MWI_LWC, and/or MWI_CTI. If the return value is 0, then the message waiting indicator is OFF.

- **getDirectoryName**

```
public abstract String getDirectoryName()
```

Returns the G3 PBX Integrated Directory Database name corresponding to this Address.

Interface com.lucent.jtapi.tsapi.LucentAddressMsgWaitingEvent

public interface **LucentAddressMsgWaitingEvent**
extends ITsapiAddressMsgWaitingEvent

This interface add Lucent−specific methods to the CallCtlAddrMessageWaitingEv interface.

Method Index

Methods

o getMessageWaitingBits

```
public abstract int getMessageWaitingBits()
```

Returns a bit−mask indicating which applications have enabled the message waiting indicator at this Address. Its value is a logical−OR combination of MWI_MCS, MWI_VOICE, MWI_PROPMGT, MWI_LWC, and/or MWI_CTI. If the return value is 0, then the message waiting indicator is OFF.

See Also:
LucentAddress

Interface com.lucent.jtapi.tsapi.LucentAgent

public interface **LucentAgent**
extends ITsapiAgent

The LucentAgent interface extends the ITsapiAgent interface.

Variable Index

Method Index

Variables

o MODE_NONE

```
public static final int MODE_NONE
```

This implies the agent's work mode is not being set.

o MODE_AUTO_IN

```
public static final int MODE_AUTO_IN
```

In this work mode the agent is put into the Agent.READY state immediately after

disconnecting from a previous call and can be delivered a new call .

o **MODE_MANUAL_IN**

```
public static final int MODE_MANUAL_IN
```

In this work mode the agent is put into the Agent.WORK_NOT_READY immediately after disconnecting from a previous call and cannot be delivered a new call .

Methods

o **setState**

```
public abstract void setState(int state,
                              int workMode) throws TsapiInvalidArgumentException, TsapiInvalidStateE
```

This method overrides Agent.setState() to add the Lucent−specific parameter workMode. It changes the state and workMode of a previously added Agent.

The post and pre conditions are as follows:

The pre−condition predicates for this method are:
1. this.getTerminal.getProvider().getState() == IN_SERVICE
2. this.getStateInfo (appropriate state and workMode)

The post−condition predicates for this method are:
1. this.getTerminal.getProvider().getState() == IN_SERVICE
2. this.getStateInfo() == state and workMode (specified as a parameter)

Parameters:
state − specifies the state this Agent should be set to. Valid states are READY, NOT_READY, WORK_READY and WORK_NOT_READY.
workMode − specifies the state this Agent should be set to. Valid workModes are MODE_AUTO_IN and MODE_MANUAL_IN.
Throws: TsapiInvalidArgumentException
At least one of the arguments passed in is not valid.
Throws: TsapiInvalidStateException
Implementation determined Agent was in an invalid state for this method.

o **getStateInfo**

```
public abstract LucentAgentStateInfo getStateInfo()
```

This returns this Agent's state and workMode.

Valid values of state returned are UNKNOWN, BUSY, READY, NOT_READY, WORK_READY, WORK_NOT_READY, LOG_IN and LOG_OUT. Valid values of

workModes are MODE_AUTO_IN and MODE_MANUAL_IN.

Interface com.lucent.jtapi.tsapi.LucentCall

public interface **LucentCall**
extends ITsapiCall, LucentCallInfo

The LucentCall interface extends ITsapiCall with Lucent−specific features. When a Provider is bound to a Lucent switch, this interface may be used to access additional capabilities.

Direct−agent calling may be used by an application to place a call to a specific logged−in ACD agent. Supervisor−assist calling may be used by an application to place a call from a logged−in ACD agent to a supervisor. These features are also available on consultation calls. These types of calls may be tracked separately by ACD measurement software in the PBX.

User−to−user information is an ISDN feature which allows end−to−end transmission of application data during call setup/teardown. This information may be a customer number, credit card number, alphanumeric digits, or a binary string. It is propagated with the call whether the call is made to a destination on the local switch or to a destination on a remote switch over PRI trunks. The switch sends the UUI in the ISDN SETUP message over the PRI trunk to establish the call. The local and the remote switch include the UUI in the alerting, connected, disconnected and route request events.

See Also:
UserToUserInfo

Method Index

Similar to the standard consult(), with the addition of Lucent−specific call parameters.

- **consultDirectAgent**(TerminalConnection, Agent, boolean, UserToUserInfo)
  Places a consultation call with a specific logged−in ACD agent.
- **consultSupervisorAssist**(TerminalConnection, ACDAddress, String, UserToUserInfo)
  Places a consultation call from a logged−in ACD agent to a supervisor's extension.

Methods

- **connect**

```
public abstract Connection[] connect(Terminal origterm,
                                     Address origaddr,
                                     String dialedDigits,
                                     boolean priorityCall,
                                     UserToUserInfo userInfo) throws TsapiResourceUnavailableExcept
```

Similar to the standard connect(), with the addition of Lucent−specific call parameters.

Parameters:

origterm − The originating Terminal for this telephone call.
origaddr − The originating Address for this telephone call.
dialedDigits − The dialable destination string for this telephone call.
priorityCall − If *true*, attempt to place a priority call
userInfo − Associate caller information, up to 32 bytes, with the call.

See Also:

UserToUserInfo

- **connectDirectAgent**

```
public abstract Connection[] connectDirectAgent(Terminal origterm,
                                                Address origaddr,
                                                Agent calledAgent,
                                                boolean priorityCall,
                                                UserToUserInfo userInfo) throws TsapiResourceUnavai
```

Places a direct call to a specific logged−in ACD agent.

Parameters:

origterm − The originating Terminal for this telephone call.
origaddr − The originating Address for this telephone call.
calledAgent − The ACD agent extension to be called.
priorityCall − If *true*, attempt to place a priority call
userInfo − Associate caller information, up to 32 bytes, with the call.

See Also:

UserToUserInfo

o **connectSupervisorAssist**

```
public abstract Connection[] connectSupervisorAssist(Agent callingAgent,
                                                     String dialedDigits,
                                                     UserToUserInfo userInfo) throws TsapiResourceU
```

Places a call from a logged−in ACD agent to a supervisor's extension.

Parameters:

callingAgent − The ACD agent extension from which to originate the call.
dialedDigits − The supervisor's extension.
userInfo − Associate caller information, up to 32 bytes, with the call.

See Also:

UserToUserInfo

o **connectPredictive**

```
public abstract Connection[] connectPredictive(Terminal originatorTerminal,
                                               Address origAddress,
                                               String dialedDigits,
                                               int connectionState,
                                               int maxRings,
                                               int answeringTreatment,
                                               int answeringEndpointType,
                                               boolean priorityCall,
                                               UserToUserInfo userInfo) throws TsapiResourceUnavaila
```

Similar to the standard connectPredictive(), with the addition of Lucent−specific call parameters.

Parameters:

originatorTerminal − The originating Terminal of the telephone call. This is optional when the originator is for example an ACDAddress.
origAddress − The originating Address of the telephone call.
dialedDigits − This must be a complete and valid telephone number.
connectionState − The application may set this to CONNECTED ALERTING, NETWORK_REACHED or NETWORK_ALERTING.
maxRings − This specifies the the number of rings that are allowed before classifying the call as no answer. The allowed range is from MIN_RINGS of 2 to MAX_RINGS of 15.
answeringTreatment − This specifies the call treatment when an answering endpoint is detected. The set includes ANSWERING_TREATMENT_PROVIDER_DEFAULT, ANSWERING_TREATMENT_DROP, ANSWERING_TREATMENT_CONNECT and ANSWERING_TREATMENT_NONE.
answeringEndpointType − This specifies the type of answering endpoint. The set includes ENDPOINT_ANSWERING_MACHINE, ENDPOINT_FAX_MACHINE, ENDPOINT_HUMAN_INTERVENTION, ENDPOINT_ANY.
priorityCall − If *true*, attempt to place a priority call

userInfo − Associate caller information, up to 32 bytes, with the call.

See Also:

UserToUserInfo

o **consult**

```
public abstract Connection[] consult(TerminalConnection termconn,
                                     String address,
                                     boolean priorityCall,
                                     UserToUserInfo userInfo) throws TsapiInvalidStateException, Tsa
```

Similar to the standard consult(), with the addition of Lucent−specific call parameters.

Parameters:

termconn − The controlling TerminalConnection for the consultation call.
address − The dialable destination string for this telephone call.
priorityCall − If *true*, attempt to place a priority call
userInfo − Associate caller information, up to 32 bytes, with the call.

See Also:

UserToUserInfo

o **consultDirectAgent**

```
public abstract Connection[] consultDirectAgent(TerminalConnection termconn,
                                                Agent calledAgent,
                                                boolean priorityCall,
                                                UserToUserInfo userInfo) throws TsapiInvalidStateExc
```

Places a consultation call with a specific logged−in ACD agent.

Parameters:

termconn − The controlling TerminalConnection for the consultation call.
calledAgent − The ACD agent extension to be called.
priorityCall − If *true*, attempt to place a priority call
userInfo − Associate caller information, up to 32 bytes, with the call.

See Also:

UserToUserInfo

o **consultSupervisorAssist**

```
public abstract Connection[] consultSupervisorAssist(TerminalConnection termconn,
                                                     ACDAddress split,
                                                     String address,
                                                     UserToUserInfo userInfo) throws TsapiInvalidSta
```

Places a consultation call from a logged−in ACD agent to a supervisor's extension.

Parameters:

termconn − The controlling TerminalConnection for the consultation call.

split − The split which the originating ACD agent is logged into.
address − The supervisor's extension.
userInfo − Associate caller information, up to 32 bytes, with the call.

See Also:

UserToUserInfo

Interface com.lucent.jtapi.tsapi.LucentCallInfo

public interface **LucentCallInfo**
extends ITsapiCallInfo

The LucentCallInfo interface provides access methods for Lucent−specific call information. These methods are implemented on the call object and on call control call events. For example, if a *CallControlCallObserver* receives a *CallCtlConnAlertingEv*, it may be cast to *LucentCallInfo* to use the *getUserToUserInfo()* method. These methods may return null if the requested data is not available.

Variable Index

Method Index

Specifies the reason for the last event sent for Connections and TerminalConnections on the Call or the Call.

- **getTrunk**()
  Returns the trunk from which the call originated.
- **getUserEnteredCode**()
  Returns call prompting digits associated with the call event.
- **getUserToUserInfo**()
  Returns user−to−user information associated with the call event.

Variables

AR_NONE

```
public static final short AR_NONE
```

AR_ANSWER_NORMAL

```
public static final short AR_ANSWER_NORMAL
```

AR_ANSWER_TIMED

```
public static final short AR_ANSWER_TIMED
```

AR_ANSWER_VOICE_ENERGY

```
public static final short AR_ANSWER_VOICE_ENERGY
```

AR_ANSWER_MACHINE_DETECTED

```
public static final short AR_ANSWER_MACHINE_DETECTED
```

AR_SIT_REORDER

```
public static final short AR_SIT_REORDER
```

AR_SIT_NO_CIRCUIT

```
public static final short AR_SIT_NO_CIRCUIT
```

AR_SIT_INTERCEPT

```
public static final short AR_SIT_INTERCEPT
```

AR_SIT_VACANT_CODE

```
public static final short AR_SIT_VACANT_CODE
```

AR_SIT_INEFFECTIVE_OTHER

```
public static final short AR_SIT_INEFFECTIVE_OTHER
```

- **AR_SIT_UNKNOWN**

```
public static final short AR_SIT_UNKNOWN
```

- **AR_IN_QUEUE**

```
public static final short AR_IN_QUEUE
```

Methods

- **getUserToUserInfo**

```
public abstract UserToUserInfo getUserToUserInfo()
```

Returns user−to−user information associated with the call event.

- **getLookaheadInfo**

```
public abstract LookaheadInfo getLookaheadInfo()
```

Returns lookahead interflow information associated with the call event.

- **getUserEnteredCode**

```
public abstract UserEnteredCode getUserEnteredCode()
```

Returns call prompting digits associated with the call event.

- **getOriginalCallInfo**

```
public abstract OriginalCallInfo getOriginalCallInfo()
```

Returns original call information associated with the call event.

- **getDistributingAddress**

```
public abstract CallCenterAddress getDistributingAddress()
```

For a connection to an AgentTerminal, getDistributingAddress returns the ACDAddress or ACDManagerAddress that was an intermediate endpoint before the call terminated at the AgentTerminal.

- **getDeliveringACDAddress**

```
public abstract ACDAddress getDeliveringACDAddress()
```

For a connection to an AgentTerminal, getDeliveringACDAddress returns the ACDAddress that this call was delivered through to the AgentTerminal.

- **getTrunk**

```
public abstract CallCenterTrunk getTrunk()
```

Returns the trunk from which the call originated.

- **getReason**

```
public abstract short getReason()
```

Specifies the reason for the last event sent for Connections and TerminalConnections on the Call or the Call.

Interface com.lucent.jtapi.tsapi.LucentConnNetworkReachedEvent

public interface **LucentConnNetworkReachedEvent**
extends ITsapiConnNetworkReachedEvent

Returns supplementary call progress information from the ISDN Progress Indicator Information Element.

Method Index

Methods

o **getNetworkProgressInfo**

```
public abstract NetworkProgressInfo getNetworkProgressInfo()
```

Get the supplementary call progress information

Interface com.lucent.jtapi.tsapi.LucentConnection

public interface **LucentConnection**
extends ITsapiConnection

The LucentConnection interface extends ITsapiConnection with Lucent−specific features. When a Provider is bound to a Lucent switch, this interface may be used to access additional capabilities.

Variable Index

Method Index

Variables

o DR_NONE

```
public static final short DR_NONE
```

o DR_CALL_CLASSIFIER

```
public static final short DR_CALL_CLASSIFIER
```

Drop a call classifier from the call..

o DR_TONE_GENERATOR

```
public static final short DR_TONE_GENERATOR
```

Drop a tone generator from the call..

Methods

o **disconnect**

```
public abstract void disconnect(short dropResource,
                                UserToUserInfo userInfo) throws TsapiPrivilegeViolationException, Ts
```

Similar to the standard disconnect(), with the addition of Lucent−specific parameters.

Parameters:

dropResource − The resource to be dropped from the call. Possible values are DR_CALL_CLASSIFIER, DR_TONE_GENERATOR, and DR_NONE.
userInfo − Associate caller information, up to 32 bytes, with the call.

See Also:

UserToUserInfo

Interface com.lucent.jtapi.tsapi.LucentProvider

public interface **LucentProvider**
extends ITsapiProvider

LucentProvider adds methods to obtain Lucent−specific switch information.

Method Index

Methods

o getTrunkGroupInfo

```
public abstract TrunkGroupInfo getTrunkGroupInfo(String trunkAccessCode)
```

Returns trunk usage information on the specified trunk group.

o getCallClassifierInfo

```
public abstract CallClassifierInfo getCallClassifierInfo()
```

Returns information on call classifier port usage.

o getSwitchDateAndTime

```
public abstract Date getSwitchDateAndTime()
```

Returns current date and time from the switch.

Interface com.lucent.jtapi.tsapi.LucentRouteSession

public interface **LucentRouteSession**
extends ITsapiRouteSession

The LucentRouteSession interface extends ITsapiRouteSession with Lucent−specific features. When a Provider is bound to a Lucent switch, this interface may be used to access additional capabilities.

Method Index

Methods

o **selectRoute**

```
public abstract void selectRoute(String routeSelected,
                                 boolean priorityCall,
                                 UserToUserInfo userInfo) throws TsapiMethodNotSupportedException
```

Similar to the standard selectRoute(), with the addition of Lucent−specific call parameters.

Parameters:
routeSelected − The selected route for this call. (Note that this is *NOT* an array.)
priorityCall − If *true*, attempt to place a priority call
userInfo − Associate caller information, up to 32 bytes, with the call.

See Also:
UserToUserInfo

o **selectRouteDirectAgent**

```
public abstract void selectRouteDirectAgent(Agent calledAgent,
                                            boolean priorityCall,
                                            UserToUserInfo userInfo) throws TsapiMethodNotSupportedE
```

Routes a direct agent call to a specific logged−in ACD agent.

Parameters:

calledAgent − The ACD agent extension to route to.
priorityCall − If *true*, attempt to place a priority call
userInfo − Associate caller information, up to 32 bytes, with the call.

See Also:

UserToUserInfo

o **selectRouteAndCollect**

```
public abstract void selectRouteAndCollect(String routeSelected,
                                           int digitsToBeCollected,
                                           int timeout,
                                           boolean priorityCall,
                                           UserToUserInfo userInfo) throws TsapiMethodNotSupportedEx
```

Routes a call and requests DTMF digit collection.

Parameters:

routeSelected − The selected route for this call. (Note that this is *NOT* an array.)
digitsToBeCollected − The number of digits to be collected (up to 24).
timeout − The number of seconds to wait (up to 63) before digit collection times out.
priorityCall − If *true*, attempt to place a priority call
userInfo − Associate caller information, up to 32 bytes, with the call.

See Also:

UserToUserInfo

o **selectRouteWithDigits**

```
public abstract void selectRouteWithDigits(String routeSelected,
                                           String digits,
                                           boolean priorityCall,
                                           UserToUserInfo userInfo) throws TsapiMethodNotSupportedEx
```

Routes a call and places digits in a dial−ahead digit buffer.

Parameters:

routeSelected − The selected route for this call. (Note that this is *NOT* an array.)
digits − A string of up to 24 characters (0−9, *, and # only) to place in the dial−ahead digit buffer.

priorityCall − If *true*, attempt to place a priority call
userInfo − Associate caller information, up to 32 bytes, with the call.

See Also:

UserToUserInfo

Interface com.lucent.jtapi.tsapi.LucentTerminal

public interface **LucentTerminal**
extends ITsapiTerminal

The LucentTerminal interface extends the ITsapiTerminal interface.

Method Index

Methods

o addAgent

```
public abstract Agent addAgent(Address agentAddress,
                               ACDAddress acdAddress,
                               int initialState,
                               int workMode,
                               String agentID,
                               String password) throws TsapiInvalidArgumentException, TsapiInvalidSt
```

This method overrides Terminal.addAgent() to add the Lucent−specific parameter workMode. It creates an Agent object, adds it to this AgentTerminal and returns the Agent object.

An Agent object represents an AgentTerminal logged into an ACDAddress.

If the getAgents() method is invoked subsequently it will return this Agent object.

The Agent can be removed from this AgentTerminal by invoking the removeAgent() method.

The pre−condition predicates for this method are:
1. this.getProvider().getState() == IN_SERVICE

The post−condition predicates for this method are:

1. this.getProvider().getState() == IN_SERVICE
2. (this.getAgents() union agent) == agent
3. agent.getStateInfo == initial state and workMode (specified as a parameter)

Parameters:

agentAddress − specifies that Address on this Terminal that this request is for, where the Terminal may support several addresses.
acdAddress − specifies the address of the ACD that the Terminal is requested to be logged in to.
initialState − is the state the specified state requested, it must be one of the states specified in the Agent object READY or NOT_READY.
workMode − specifies the state this Agent should be set to. Valid workModes are MODE_AUTO_IN and MODE_MANUAL_IN.
agentID − is the Agents ID.
password − is the Agents password.

Returns:

An Agent object representing the association between this AgentTerminal and the ACDAddress specified in the request.

Throws: TsapiInvalidArgumentException

At least one of the arguments provided is not valid.

Throws: TsapiInvalidStateException

Implementation determined AgentTerminal was in an invalid state for this method.

o **getDirectoryName**

```
public abstract String getDirectoryName()
```

Return Directory name of this Terminal.

Interface com.lucent.jtapi.tsapi.LucentTerminalConnection

public interface **LucentTerminalConnection**
extends ITsapiTerminalConnection

The LucentTerminalConnection interface extends ITsapiTerminalConnection with Lucent−specific features. When a Provider is bound to a Lucent switch, this interface may be used to access additional capabilities.

Variable Index

Method Index

Variables

o DR_NONE

```
public static final short DR_NONE
```

o DR_CALL_CLASSIFIER

```
public static final short DR_CALL_CLASSIFIER
```

Drop a call classifier from the call..

o DR_TONE_GENERATOR

```
public static final short DR_TONE_GENERATOR
```

Drop a tone generator from the call..

Methods

o **leave**

```
public abstract void leave(short dropResource,
                           UserToUserInfo userInfo) throws TsapiInvalidStateException, TsapiMethodNo
```

Similar to the standard leave(), with the addition of Lucent−specific parameters.

Parameters:

dropResource − The resource to be dropped from the call. Possible values are DR_CALL_CLASSIFIER, DR_TONE_GENERATOR, and DR_NONE.
userInfo − Associate caller information, up to 32 bytes, with the call.

See Also:

disconnect, UserToUserInfo

Class com.lucent.jtapi.tsapi.CallClassifierInfo

```
java.lang.Object
   |
   +----com.lucent.jtapi.tsapi.ASN1
           |
           +----com.lucent.jtapi.tsapi.ASNSequence
                   |
                   +----com.lucent.jtapi.tsapi.LucentPrivateData
                           |
                           +----com.lucent.jtapi.tsapi.CallClassifierInfo
```

---

public final class **CallClassifierInfo**
extends LucentPrivateData

Provides information on call classifier port usage.

Variable Index

Variables

o numAvailPorts

```
public int numAvailPorts
```

The number of available call classifier ports.

o numInUsePorts

```
public int numInUsePorts
```

The number of in−use call classifier ports.

Class com.lucent.jtapi.tsapi.LookaheadInfo

```
java.lang.Object
   |
   +----com.lucent.jtapi.tsapi.ASN1
            |
            +----com.lucent.jtapi.tsapi.ASNSequence
                     |
                     +----com.lucent.jtapi.tsapi.LucentPrivateData
                              |
                              +----com.lucent.jtapi.tsapi.LookaheadInfo
```

---

public final class **LookaheadInfo**
extends LucentPrivateData

Lookahead interflow is a G3 switch feature that routes some of the incoming calls from one switch to another so that they can be handled more efficiently and will not be lost. The lookahead interflow information is provided by the switch that overflows the call. The routing server application may use the lookahead interflow information to determine the destination of the call.

Variable Index

Method Index

Priority of the interflowed call.

- **getSeconds**()
  Gets the 'seconds' part of the event timestamp.
- **getSourceVDN**()
  Returns the address of the VDN which overflowed the call.
- **getType**()
  Type of interflow.

Variables

LAI_ALL_INTERFLOW

```
public static final short LAI_ALL_INTERFLOW
```

LAI_THRESHOLD_INTERFLOW

```
public static final short LAI_THRESHOLD_INTERFLOW
```

LAI_VECTORING_INTERFLOW

```
public static final short LAI_VECTORING_INTERFLOW
```

LAI_NOT_IN_QUEUE

```
public static final short LAI_NOT_IN_QUEUE
```

LAI_LOW

```
public static final short LAI_LOW
```

LAI_MEDIUM

```
public static final short LAI_MEDIUM
```

LAI_HIGH

```
public static final short LAI_HIGH
```

LAI_TOP

```
public static final short LAI_TOP
```

Methods

getType

```
public short getType()
```

Type of interflow. Possible values are LAI_ALL_INTERFLOW, LAI_THRESHOLD_INTERFLOW, and LAI_VECTORING_INTERFLOW.

- **getPriority**

```
public short getPriority()
```

Priority of the interflowed call. Possible values are LAI_NOT_IN_QUEUE, LAI_LOW, LAI_MEDIUM, LAI_HIGH, and LAI_TOP.

- **getHours**

```
public int getHours()
```

Gets the 'hours' part of the event timestamp.

- **getMinutes**

```
public int getMinutes()
```

Gets the 'minutes' part of the event timestamp.

- **getSeconds**

```
public int getSeconds()
```

Gets the 'seconds' part of the event timestamp.

- **getSourceVDN**

```
public ACDManagerAddress getSourceVDN()
```

Returns the address of the VDN which overflowed the call.

Class com.lucent.jtapi.tsapi.LucentAgentStateInfo

```
java.lang.Object
   |
   +----com.lucent.jtapi.tsapi.LucentAgentStateInfo
```

---

public final class **LucentAgentStateInfo**
extends Object

This is the object that is returned by the query getStateInfo in LucentAgent. It returns both the state and workMode for the Agent.

See LucentAgent for details.

Variable Index

Constructor Index

Variables

o **state**

```
public int state
```

State of Agent.

o **workMode**

```
public int workMode
```

Work Mode for Agent.

Constructors

o LucentAgentStateInfo

```
public LucentAgentStateInfo(int _state,
                            int _workMode)
```

Public Constructor.

Class com.lucent.jtapi.tsapi.NetworkProgressInfo

```
java.lang.Object
   |
   +----com.lucent.jtapi.tsapi.ASN1
           |
           +----com.lucent.jtapi.tsapi.ASNSequence
                   |
                   +----com.lucent.jtapi.tsapi.LucentPrivateData
                           |
                           +----com.lucent.jtapi.tsapi.NetworkProgressInfo
```

---

public final class **NetworkProgressInfo**
extends LucentPrivateData

Contains supplementary call progress information from the ISDN Progress Indicator Information Element.

Variable Index

Variables

o PL_USER

```
public static final short PL_USER
```

o PL_PUB_LOCAL

```
public static final short PL_PUB_LOCAL
```

o PL_PUB_REMOTE

```
public static final short PL_PUB_REMOTE
```

o PL_PRIV_REMOTE

```
public static final short PL_PRIV_REMOTE
```

o PD_CALL_OFF_ISDN

```
public static final short PD_CALL_OFF_ISDN
```

o PD_DEST_NOT_ISDN

```
public static final short PD_DEST_NOT_ISDN
```

o PD_ORIG_NOT_ISDN

```
public static final short PD_ORIG_NOT_ISDN
```

o PD_CALL_ON_ISDN

```
public static final short PD_CALL_ON_ISDN
```

o PD_INBAND

```
public static final short PD_INBAND
```

o progressLocation

```
public short progressLocation
```

Specifies the progress location in a Progress Indicator Information Element from the PRI network. Possible values are PL_USER, PL_PUB_LOCAL, PL_PUB_REMOTE, PL_PRIV_REMOTE

o progressDescription

```
public short progressDescription
```

Specifies the progress description in a Progress Indicator Information Element from the PRI network. Possible values are PD_CALL_OFF_ISDN, PD_DEST_NOT_ISDN, PD_ORIG_NOT_ISDN, PD_CALL_ON_ISDN, PD_INBAND

Class com.lucent.jtapi.tsapi.OriginalCallInfo

```
java.lang.Object
   |
   +----com.lucent.jtapi.tsapi.ASN1
            |
            +----com.lucent.jtapi.tsapi.ASNSequence
                     |
                     +----com.lucent.jtapi.tsapi.LucentPrivateData
                              |
                              +----com.lucent.jtapi.tsapi.OriginalCallInfo
```

---

public final class **OriginalCallInfo**
extends LucentPrivateData

Original Call Information is made available in conjunction with the consult() service. It is provided in event reports to observers of the consulted party and contains information about the original call.

Variable Index

Method Index

Get the original collected digits for this call.

- **getUserToUserInfo**()
  Get the original user−to−user information for this call.

Variables

OR_CONSULTATION

```
public static final short OR_CONSULTATION
```

OR_CONFERENCED

```
public static final short OR_CONFERENCED
```

OR_TRANSFERRED

```
public static final short OR_TRANSFERRED
```

OR_NEW_CALL

```
public static final short OR_NEW_CALL
```

Methods

getReason

```
public short getReason()
```

Get the reason code for this OriginalCallInfo. Possible values are OR_CONSULTATION, OR_CONFERENCED, OR_TRANSFERRED, and OR_NEW_CALL.

getCallingDevice

```
public Address getCallingDevice()
```

Get the original calling device for this call.

getCalledDevice

```
public Address getCalledDevice()
```

Get the original called device for this call.

getTrunk

```
public CallCenterTrunk getTrunk()
```

Get the original trunk device for this call.

- **getUserToUserInfo**

```
public UserToUserInfo getUserToUserInfo()
```

Get the original user−to−user information for this call.

- **getLookaheadInfo**

```
public LookaheadInfo getLookaheadInfo()
```

Get the original lookahead information for this call.

- **getUserEnteredCode**

```
public UserEnteredCode getUserEnteredCode()
```

Get the original collected digits for this call.

Class com.lucent.jtapi.tsapi.TrunkGroupInfo

```
java.lang.Object
   |
   +----com.lucent.jtapi.tsapi.ASN1
            |
            +----com.lucent.jtapi.tsapi.ASNSequence
                     |
                     +----com.lucent.jtapi.tsapi.LucentPrivateData
                              |
                              +----com.lucent.jtapi.tsapi.TrunkGroupInfo
```

---

public final class **TrunkGroupInfo**
extends LucentPrivateData

Provides information on trunk group usage.

Variable Index

Variables

o idleTrunks

```
public int idleTrunks
```

The number of idle trunks.

o usedTrunks

```
public int usedTrunks
```

The number of in−use trunks.

Class com.lucent.jtapi.tsapi.UserEnteredCode

```
java.lang.Object
   |
   +----com.lucent.jtapi.tsapi.ASN1
           |
           +----com.lucent.jtapi.tsapi.ASNSequence
                   |
                   +----com.lucent.jtapi.tsapi.LucentPrivateData
                           |
                           +----com.lucent.jtapi.tsapi.UserEnteredCode
```

---

public final class **UserEnteredCode**
extends LucentPrivateData

Contains the code/digits that may have been entered by the caller through the G3 call prompting feature or the collected digits feature.

The following are necessary steps for setting up VDNs, simple vector steps and CallObservers in order for a client application to receive UECs from the switch.

1. Administer a VDN and a vector on the G3 switch with collect digits step and route command to a second VDN. See Call Scenario 1 and 2 below.

   The purpose of this VDN is to collect UEC, but it will not report the UEC to the PBX driver, even if the VDN is observed. The route command must redirect the call to a second VDN. *The first VDN doesn´t have to be observed by any client application.*

2. Administer a second VDN and vector to receive the redirected call from the first VDN.

   The purpose of this second VDN is to report the UEC to the PBX driver. *Thus a CallObserver must be placed on the second VDN*. This VDN should redirect the call to its destination. The destination can be a station extension, an ACD split, or another VDN.

   If the destination is a station extension and there is a CallObserver on that Address, call events for that observer will contain the UEC collected by the first VDN.

   If the destination is an ACD split and there is a CallObserver on an agent station

in the split, call events for that observer will contain the UEC collected by the first VDN.

If the destination is a VDN, UEC is *NOT* delivered to observers of that VDN.

If multiple UECs are collected by multiple VDNs in call processing, only the most recently collected UEC is reported.

Limitations

1. An observed VDN only reports the UEC it receives (UEC collected in a previous VDN). It will not report UEC it collects or UEC collected after the call is redirected from the VDN.

2. A CallObserver on a station receives only the UEC that is received by the VDN that redirects the call to the station, provided that the VDN is observed (see Call Scenario 2).

Call Scenario 1:

VDN 24101 is mapped to vector 1 and vector 1 has the following steps:

1. collect 16 digits after announcement extension 1000
2. route to 24102
3. stop

VDN 24102 is mapped to vector 2 and vector 2 has the following steps:

1. route to 24103
2. stop

where 24103 is a station extension.

When a call arrives on VDN 24101, the caller will hear the announcement and the switch will wait for the caller to enter 16 digits. After the 16 digits are collected in time (if the collect digits step is timed out, next step is executed), the call is routed to VDN 24102. The VDN 24102 routes the call to station 24103.

A CallObserver on VDN 24101 will *NOT* receive UEC.

If there is a CallObserver on VDN 24102, the 16 digits collected by VDN 24101 will be reported to that observer. VDN 24101 observing is not required for VDN 24102 to receive UEC collected by VDN 24101.

If there are CallObservers on VDN 24102 and station 24103, the 16 digits collected by VDN 24101 will be reported to those observers.

Whether the station 24103 is observed or not, the 16 digits will *NOT* be reported to the VDN 24102 observer when call is delivered to station 24103.

Call Scenario 2:

VN 24201 is mapped to vector 11 and vector 11 has the following steps:

1. collect 10 digits after announcement extension 2000
2. route to 24202
3. stop

VDN 24202 is mapped to vector 12 and vector 12 has the following steps:

1. collect 16 digits after announcement extension 3000
2. route to 24203
3. stop

VDN 24203 is mapped to vector 13 and vector 13 has the following steps:

1. queue to main split 2 priority m
2. stop

   where split 2 is a vector controlled ACD split that has agent extensions 24301, 24302, 24303.

When a call arrives on VDN 24201, the caller will hear an announcement and the switch will wait for the caller to enter 10 digits. After the 10 digits are collected in time, the call is routed to VDN 24202. When the call arrives on VDN 24202, the caller will hear an announcement and the switch will wait for the caller to enter 16 digits. After the 16 digits are collected in time, the call is routed to VDN 24203. The VDN 24203 queues the call to ACD Split 2. If the agent at station 24301 is available, the call is sent to station 24301.

A CallObserver on VDN 24201 will *NOT* receive UEC.

If there is a CallObserver on VDN 24102, the 10 digits collected by VDN 24201 will be reported to that observer.

If there is a CallObserver on VDN 24203, the 16 digits collected by VDN 24202 will be reported to that observer. However, the 10 digits collected by VDN 24201 will *NOT* be reported to that observer. An observer receives only the most recent UEC.

If VDN 24202 and VDN 24203 and station 24301 are all observed, only the 16 digits collected by VDN 24202 will be reported to the station 24301 observer. A station observer will receive the UEC that is received by the VDN that redirects calls to the station.

NOTE: In order to receive the UEC at a station observer, the VDN that receives the UEC and redirects calls to the station must be observed. For example, if VDN 24203 is *NOT* observed by any client, an observer on station 24301 will *NOT* receive the 16 digits collected by VDN 24202.

Variable Index

Method Index

Variables

o **UE_ANY**

```
public static final short UE_ANY
```

o **UE_LOGIN_DIGITS**

```
public static final short UE_LOGIN_DIGITS
```

o **UE_CALL_PROMPTER**

```
public static final short UE_CALL_PROMPTER
```

o **UE_DATA_BASE_PROVIDED**

```
public static final short UE_DATA_BASE_PROVIDED
```

o **UE_TONE_DETECTOR**

```
public static final short UE_TONE_DETECTOR
```

- **UE_COLLECT**

```
public static final short UE_COLLECT
```

- **UE_ENTERED**

```
public static final short UE_ENTERED
```

Methods

- **getType**

```
public short getType()
```

Returns the type of digits collected

- **getIndicator**

```
public short getIndicator()
```

Returns UE_COLLECT or UE_ENTERED

- **getDigits**

```
public String getDigits()
```

Returns the collected digits

- **getCollectVDN**

```
public ACDManagerAddress getCollectVDN()
```

Returns the ACDManagerAddress of the VDN which collected the digits

---

Class com.lucent.jtapi.tsapi.UserToUserInfo

```
java.lang.Object
   |
   +----com.lucent.jtapi.tsapi.ASN1
           |
           +----com.lucent.jtapi.tsapi.ASNSequence
                   |
                   +----com.lucent.jtapi.tsapi.LucentPrivateData
                           |
                           +----com.lucent.jtapi.tsapi.UserToUserInfo
```

---

public final class **UserToUserInfo**
extends LucentPrivateData

User−to−user information is an ISDN feature which allows end−to−end transmission of application data during call setup/teardown. This information may be a customer number, credit card number, alphanumeric digits, or a binary string. It is propagated with the call whether the call is made to a destination on the local switch or to a destination on a remote switch over PRI trunks. The switch sends the UUI in the ISDN SETUP message over the PRI trunk to establish the call. The local and the remote switch include the UUI in the alerting, connected, disconnected and route request events.

Constructor Index

Method Index

Constructors

o **UserToUserInfo**

```
public UserToUserInfo(String _data)
```

construct a UserToUserInfo object from an ASCII string

o **UserToUserInfo**

```
public UserToUserInfo(byte _data[])
```

construct a UserToUserInfo object from a byte array

Methods

o **getString**

```
public String getString()
```

return user−to−user info as an ASCII string

o **getBytes**

```
public byte[] getBytes()
```

return user−to−user info as a (binary) byte array

o **isAscii**

```
public boolean isAscii()
```

query whether sender encoded UUI as ASCII or binary

Using Telephony Services Private Data Extensions to JTAPI

4

Contents

Using Telephony Services Private Data Extensions to JTAPI

4

Using Telephony Services Private Data Extensions

NOTE:
This chapter describes non-standard additions to JTAPI. This package is available only from the PassageWay Telephony Services implementation of JTAPI and is not available from any other implementation of JTAPI.

This chapter contains the extensions that support Telephony Services implementation of JTAPI for the private data mechanism for non-DEFINITY switches and their associated drivers.

Who Should Be Using These Extensions?

An independent switch vendor who is using the JTAPI private data programming environment to develop a private data package for non-DEFINITY switches, or an application programmer who is using or interpreting private data in a raw form, without an intermediate private data package. (An example of an intermediate private data package that allows programmers to access private data via Java interfaces rather than through raw private data bytes is contained in Chapter 3, “Using Telephony Services DEFINITY-Specific Extensions.) It is assumed that this individual has a familiarity with the Java programming language, JTAPI, Lucent Technologies Telephony Services Application Programmer’s Interface (TSAPI) and its private data mechanism.

NOTE:
If you are an application programmer who is using JTAPI to develop applications for any switch for which there is a PassageWay Telephony Services driver, ignore this chapter and refer to Chapter 1, “Telephony Services Implementation of JTAPI for All Switches and the DEFINITY Switch.” If you want additional TSAPI-specific information about the JTAPI exceptions that are not accessible through standard JTAPI, refer to Chapter 2, “ Using Telephony Services Extensions to JTAPI Exceptions.”

NOTE:
If you are an application programmer who is using JTAPI to develop applications for the DEFINITY switch, ignore this chapter and refer to Chapter 1, “Telephony Services Implementation of JTAPI for All Switches and the DEFINITY Switch.” If you want to take advantage of DEFINITY-specific features that are not accessible through standard JTAPI, refer to Chapter 3, “Telephony Services DEFINITY-Specific Extensions to JTAPI.”

How Should the Extensions be Used?

The private data extensions to JTAPI assist independent switch vendors in the creation of a private data package for non-DEFINITY switches, or allow application programmers to use or interpret private data when they are supplied with private data in its raw form (i.e., without an intermediate private data package.)

The following sections describe guidelines for using or interpreting private data when it is supplied in its raw form.

Initialization of Private Data

In order to use or interpret private data from a particular vendor, the application must specify the vendor name and the version of the private data that is to be used. The particular format of the name and version strings used is supplied by the vendor.

The specification of the vendor name and the version of the private data must be done after the application creates a JtapiPeer but before it creates the Provider. The ITsapiPeer.addVendor() method allows vendor names and versions to be specified to the application. For example, if a JtapiPeer has been created (called peer) which is an instance of ITsapiPeer, then:

```
((ITsapiPeer)peer).addVendor("Brand X", "1-3")
```

indicates that the application knows how to interpret private data from vendor “Brand X” as well as versions 1, 2, and 3 of that private data.

If the application supports private data produced by multiple vendors, the application may call addVendor() multiple times before receiving the Provider.

When a String containing the vendor name and version is passed to JtapiPeer.getProvider(), a particular Provider will be connected to a single vendor delivering one particular version of private data. The application determines the connected vendor and version by executing the ITsapiProvider.getVendor() and ITsapiProvider.getVendorVersion() methods. Once a particular vendor and version is associated with a particular Provider, this association will not change for the life of the Provider. If the application wants a different Provider, the application must call ITsapiPeer.addVendor() again.

Using TsapiPrivate as a JTAPI Private Data Object

Where JTAPI specifies that a private data Object is to be passed in as an argument to a method, this implementation of JTAPI requires the Object to be an instance of TsapiPrivate. Where JTAPI specifies that a private data Object is to be returned from a method, in this implementation, the returned Object is always an instance of TsapiPrivate.

When constructing a TsapiPrivate object to be used with the sendPrivateData() methods, waitForResponse must be set so that the appropriate action is taken.

- A value of true indicates that the implementation should block sendPrivateData() until a response is received from the switch. This reponse will be passed back to the application as the return code from sendPrivateData(). This is equivalent to the TSAPI request cstaEscapeService().

- A value of false indicates that the implementation should return immediately (with a null) from sendPrivateData(), without waiting for a response from the switch. This is equivalent to the TSAPI request cstaSendPrivateEvent().

- When a TsapiPrivate object is passed as an argument to a setPrivateData() method, the waitForResponse flag is ignored.

Converting TSAPI Constructs to JTAPI Objects

Since private data, by its nature, cannot be interpreted by the implementation, raw TSAPI constructs may be exposed. The ITsapiProviderPrivate interface defines methods that allow raw TSAPI constructs to be converted into their JTAPI equivalents. The following table lists the raw TSAPI constructs that may be converted into their JTAPI equivalents. It lists the TSAPI constructs, the Java version (the Java class) of the TSAPI constructs, the JTAPI objects to which they are converted, and the method to be used for the conversion.

TSAPI Construct	Java Class	JTAPI Object	Conversion Method in TsapiProviderPrivate
ExtendedDeviceID_t	ExtendedDeviceID	Address	getAddress()
ExtendedDeviceID_t	ExtendedDeviceID	Terminal	getTerminal()
ConnectionID_t	ConnectionID	Connection	getConnection()
ConnectionID_t	ConnectionID	TerminalConnection	getTerminalConnection()
callID (field in a ConnectionID_t)	int	Call	getCall()

NOTE:
TSAPI constructs such as DeviceID may be converted to JTAPI objects via standard JTAPI methods such as Provider.getAddress(String) and Provider.getTerminal(String).

Converting JTAPI Objects to TSAPI Constructs

Just as receiving TSAPI private data may expose raw TSAPI constructs, sending TSAPI private data may require raw TSAPI constructs as well. The ITsapiConnIDPrivate and ITsapiRoutePrivate interfaces have been defined to retrieve TSAPI constructs from JTAPI objects.

The following table lists the JTAPI objects that may be converted into their equivalent TSAPI constructs. It lists the JTAPI objects, the TSAPI constructs to which they are converted, the Java version (the Java class) of the TSAPI constructs, and the method to be used for the conversion.

JTAPI Object	TSAPI Construct	Java Class	Conversion Method
Connection	ConnectionID_t	ConnectionID	ITsapiConnIDPrivate. getTsapiConnectionID()
TerminalConnection	ConnectionID_t	ConnectionID	ITsapiConnIDPrivate. getTsapiConnectionID()
RouteSession	RouteRegisterReqID_t	int	ITsapiRoutePrivate. getRouteRegisterID()
RouteSession	RouteCrossRefID_t	int	ITsapiRoutePrivate. getRouteCrossRefID()

package com.lucent.jtapi.tsapi

Interface Index

- ITsapiConnIDPrivate
- ITsapiPeer
- ITsapiProvider
- ITsapiProviderPrivate
- ITsapiRoutePrivate

Class Index

- ConnectionID
- ExtendedDeviceID
- TsapiPrivate

Interface com.lucent.jtapi.tsapi.ITsapiConnIDPrivate

public interface **ITsapiConnIDPrivate**

ITsapiConnIDPrivate lets you retrieve TSAPI information associated with a JTAPI Connection or TerminalConnection.

See Also:
ConnectionID

Method Index

Methods

o getTsapiConnectionID

```
public abstract ConnectionID getTsapiConnectionID()
```

Retrieves the TSAPI ConnectionID associated with a JTAPI Connection or TerminalConnection.

See Also:
ConnectionID

Interface com.lucent.jtapi.tsapi.ITsapiPeer

public interface **ITsapiPeer**
extends JtapiPeer

ITsapiPeer extends JtapiPeer to allow applications a mechanism to specify the vendor(s) they want to negotiate private data with.

Method Index

Methods

o **addVendor**

```
public abstract void addVendor(String vendorName,
                               String versions)
```

This method can be used to set the vendor the application wants to exchange private data with. The interfaces for private data are in javax.telephony.privatedata.

This method should be invoked before the application invokes getProvider().

To set multiple vendors an application must invoke this method multiple times.

Interface com.lucent.jtapi.tsapi.ITsapiProvider

public interface **ITsapiProvider**
extends Provider, CallCenterProvider

ITsapiProvider adds methods to obtain vendor−specific version information.

See Also:
addVendor

Method Index

Methods

o getVendor

```
public abstract String getVendor()
```

Returns the private data vendor name.

o getVendorVersion

```
public abstract byte[] getVendorVersion()
```

Returns the negotiated vendor private data version.

Interface com.lucent.jtapi.tsapi.ITsapiProviderPrivate

public interface **ITsapiProviderPrivate**

ITsapiProviderPrivate lets you retrieve or create JTAPI objects from TSAPI constructs.

See Also:
ConnectionID, ExtendedDeviceID

Method Index

Methods

o getCall

```
public abstract Call getCall(int callID)
```

Returns a JTAPI Call associated with a TSAPI Call ID.

Parameters:
callID − The Call ID.

o getAddress

```
public abstract Address getAddress(ExtendedDeviceID deviceID)
```

Returns a JTAPI Address associated with a TSAPI Extended Device ID.

Parameters:

deviceID − The Extended Device ID.

See Also:

ExtendedDeviceID

o **getTerminal**

```
public abstract Terminal getTerminal(ExtendedDeviceID deviceID)
```

Returns a JTAPI Terminal associated with a TSAPI Extended Device ID.

Parameters:

deviceID − The Extended Device ID.

See Also:

ExtendedDeviceID

o **getConnection**

```
public abstract Connection getConnection(ConnectionID connID,
                                         Address address)
```

Returns a JTAPI Connection associated with a TSAPI Connection ID and the specified JTAPI Address.

Parameters:

connID − The Connection ID.
address − The Address to associate with the Connection to be created.

See Also:

ConnectionID

o **getTerminalConnection**

```
public abstract TerminalConnection getTerminalConnection(ConnectionID connID,
                                                         Terminal terminal)
```

Returns a JTAPI TerminalConnection associated with a TSAPI Connection ID and the specified JTAPI Terminal.

Parameters:

connID − The Connection ID.
terminal − The Terminal to associate with the TerminalConnection to be created.

See Also:

ConnectionID

Interface com.lucent.jtapi.tsapi.ITsapiRoutePrivate

public interface **ITsapiRoutePrivate**

ITsapiRoutePrivate lets you retrieve TSAPI information associated with a JTAPI Route Session.

Method Index

Methods

o getRouteRegisterID

```
public abstract int getRouteRegisterID()
```

Retrieves the TSAPI RouteRegisterID with a JTAPI Route Session.

o getRouteCrossRefID

```
public abstract int getRouteCrossRefID()
```

Retrieves the TSAPI RouteCrossReferenceID associated with a JTAPI Route Session.

Class com.lucent.jtapi.tsapi.ConnectionID

```
java.lang.Object
   |
   +----com.lucent.jtapi.tsapi.ASN1
           |
           +----com.lucent.jtapi.tsapi.ASNSequence
                   |
                   +----com.lucent.jtapi.tsapi.ConnectionID
```

---

public final class **ConnectionID**
extends ASNSequence

Variable Index

Method Index

Variables

o STATIC_ID

```
public static final short STATIC_ID
```

o DYNAMIC_ID

```
public static final short DYNAMIC_ID
```

Methods

o **hashCode**

```
public int hashCode()
```

Overrides:
hashCode in class Object

o **equals**

```
public boolean equals(Object anObject)
```

Overrides:
equals in class Object

o **toString**

```
public String toString()
```

Overrides:
toString in class Object

Class com.lucent.jtapi.tsapi.ExtendedDeviceID

```
java.lang.Object
   |
   +----com.lucent.jtapi.tsapi.ASN1
           |
           +----com.lucent.jtapi.tsapi.ASNSequence
                   |
                   +----com.lucent.jtapi.tsapi.ExtendedDeviceID
```

---

public final class **ExtendedDeviceID**
extends ASNSequence

A TSAPI Extended Device ID. This class should be used for interpretation of TSAPI private data. Once an Extended Device ID has been constructed from TSAPI private data, a JTAPI Address or Terminal object should be created using the appropriate method in ITsapiProviderPrivate.

See Also:
ITsapiProviderPrivate

Variable Index

- **EXPLICIT_PUBLIC_ABBREVIATED**
  Device ID Type.
- **EXPLICIT_PUBLIC_INTERNATIONAL**
  Device ID Type.
- **EXPLICIT_PUBLIC_NATIONAL**
  Device ID Type.
- **EXPLICIT_PUBLIC_NETWORK_SPECIFIC**
  Device ID Type.
- **EXPLICIT_PUBLIC_SUBSCRIBER**
  Device ID Type.
- **EXPLICIT_PUBLIC_UNKNOWN**
  Device ID Type.
- **ID_NOT_KNOWN**
  Device ID Status of *ID_NOT_KNOWN* indicates the Device ID is not known.
- **ID_NOT_REQUIRED**
  Device ID Status of *ID_NOT_REQUIRED* indicates the Device ID is not required.
- **ID_PROVIDED**
  Device ID Status of *ID_PROVIDED* indicates the Device ID is valid
- **IMPLICIT_PRIVATE**
  Device ID Type.
- **IMPLICIT_PUBLIC**
  Device ID Type.
- **OTHER_PLAN**
  Device ID Type.
- **TRUNK_GROUP_IDENTIFIER**
  Device ID Type.
- **TRUNK_IDENTIFIER**
  Device ID Type.

Constructor Index

- **ExtendedDeviceID**(String, short, short)
  Construct an ExtendedDeviceID.

Variables

DEVICE_IDENTIFIER

```
public static final short DEVICE_IDENTIFIER
```

Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

IMPLICIT_PUBLIC

```
public static final short IMPLICIT_PUBLIC
```

Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **EXPLICIT_PUBLIC_UNKNOWN**

```
public static final short EXPLICIT_PUBLIC_UNKNOWN
```

  Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **EXPLICIT_PUBLIC_INTERNATIONAL**

```
public static final short EXPLICIT_PUBLIC_INTERNATIONAL
```

  Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **EXPLICIT_PUBLIC_NATIONAL**

```
public static final short EXPLICIT_PUBLIC_NATIONAL
```

  Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **EXPLICIT_PUBLIC_NETWORK_SPECIFIC**

```
public static final short EXPLICIT_PUBLIC_NETWORK_SPECIFIC
```

  Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **EXPLICIT_PUBLIC_SUBSCRIBER**

```
public static final short EXPLICIT_PUBLIC_SUBSCRIBER
```

  Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **EXPLICIT_PUBLIC_ABBREVIATED**

```
public static final short EXPLICIT_PUBLIC_ABBREVIATED
```

  Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **IMPLICIT_PRIVATE**

```
public static final short IMPLICIT_PRIVATE
```

  Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **EXPLICIT_PRIVATE_UNKNOWN**

```
public static final short EXPLICIT_PRIVATE_UNKNOWN
```

  Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **EXPLICIT_PRIVATE_LEVEL3_REGIONAL_NUMBER**

```
public static final short EXPLICIT_PRIVATE_LEVEL3_REGIONAL_NUMBER
```

Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **EXPLICIT_PRIVATE_LEVEL2_REGIONAL_NUMBER**

```
public static final short EXPLICIT_PRIVATE_LEVEL2_REGIONAL_NUMBER
```

Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **EXPLICIT_PRIVATE_LEVEL1_REGIONAL_NUMBER**

```
public static final short EXPLICIT_PRIVATE_LEVEL1_REGIONAL_NUMBER
```

Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **EXPLICIT_PRIVATE_PTN_SPECIFIC_NUMBER**

```
public static final short EXPLICIT_PRIVATE_PTN_SPECIFIC_NUMBER
```

Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **EXPLICIT_PRIVATE_LOCAL_NUMBER**

```
public static final short EXPLICIT_PRIVATE_LOCAL_NUMBER
```

Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **EXPLICIT_PRIVATE_ABBREVIATED**

```
public static final short EXPLICIT_PRIVATE_ABBREVIATED
```

Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **OTHER_PLAN**

```
public static final short OTHER_PLAN
```

Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

- **TRUNK_IDENTIFIER**

```
public static final short TRUNK_IDENTIFIER
```

Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

o **TRUNK_GROUP_IDENTIFIER**

```
public static final short TRUNK_GROUP_IDENTIFIER
```

Device ID Type. Ignored if Device ID Status is not *ID_PROVIDED*

o **ID_PROVIDED**

```
public static final short ID_PROVIDED
```

Device ID Status of *ID_PROVIDED* indicates the Device ID is valid

o **ID_NOT_KNOWN**

```
public static final short ID_NOT_KNOWN
```

Device ID Status of *ID_NOT_KNOWN* indicates the Device ID is not known. The Device ID and Device Type fields are ignored.

o **ID_NOT_REQUIRED**

```
public static final short ID_NOT_REQUIRED
```

Device ID Status of *ID_NOT_REQUIRED* indicates the Device ID is not required. The Device ID and Device Type fields are ignored.

Constructors

o **ExtendedDeviceID**

```
public ExtendedDeviceID(String _deviceID,
                        short _deviceIDType,
                        short _deviceIDStatus)
```

Construct an ExtendedDeviceID.

Parameters:

_deviceID − The Device ID.
_deviceIDType − The Device ID Type.
_deviceIDStatus − The status of the Device ID (*ID_PROVIDED*, *ID_NOT_KNOWN*, *ID_NOT_REQUIRED*).

Class com.lucent.jtapi.tsapi.TsapiPrivate

```
java.lang.Object
   |
   +----com.lucent.jtapi.tsapi.TsapiPrivate
```

---

public final class **TsapiPrivate**
extends Object

The TsapiPrivate object is used to pass vendor−specific information between an application and the service provider, via the JTAPI private data interfaces. Where JTAPI specifies that a private data Object is to be passed in as an argument to a method, this implementation requires the Object to be an instance of TsapiPrivate. Where JTAPI specifies that a private data Object is to be returned from a method, in this implementation the returned Object is always an instance of TsapiPrivate.

An application must first use the *ITsapiPeer.addVendor()* method so that when a provider is created it may negotiate the version of private data to be used.

See Also:
addVendor

Variable Index

Constructor Index

Method Index

o **getData**()
Return the byte array containing the raw private data.

Variables

o **vendor**

```
public String vendor
```

o **data**

```
public byte data[]
```

o **tsType**

```
public int tsType
```

Constructors

o **TsapiPrivate**

```
public TsapiPrivate(byte _data[])
```

Construct a TSAPI private data object. This version of the constructor should be used when this object will be passed in a *setPrivateData()* method OR when *sendPrivateData()* can return immediately (with a null) without waiting for a response from the switch (this is equivalent to the TSAPI request cstaSendPrivateEvent()).

o **TsapiPrivate**

```
public TsapiPrivate(byte _data[],
                    boolean waitForResponse)
```

Construct a TSAPI private data object. If this object is to be used with the *sendPrivateData()* methods, *waitForResponse* must be set so that the appropriate action is taken. *true* indicates that the implementation should block in *sendPrivateData()* until a response is received from the switch. This response will be passed back to the application as the return code from *sendPrivateData()*. This is equivalent to the TSAPI request cstaEscapeService(). *false* indicates that the implementation should return immediately (with a null) from *sendPrivateData()* without waiting for a response from the switch. This is equivalent to the TSAPI request cstaSendPrivateEvent(). When a TSAPI private data object is passed as an argument to a *setPrivateData()* method, the waitForResponse flag is ignored

Methods

o **getData**

```
public byte[] getData()
```

Return the byte array containing the raw private data.

Telephony Services Implementation of JTAPI for Private Data

A

Contents

Telephony Services Implementation of JTAPI for Private Data

A

This appendix describes the level of support the PassageWay Telephony Services implementation of JTAPI provides for the private data mechanism for non-DEFINITY switches and their associated drivers. It contains the mappings of Telephony Services Application Programming Interface (TSAPI) requests with the JTAPI interfaces and associated methods, along with the mappings between TSAPI and typically occurring JTAPI events.

It is suggested reading for an independent switch vendor who is using the JTAPI private data programming environment to develop a private data package for non-DEFINITY switches, or an application programmer who is using or interpreting private data in a raw form, without an intermediate private data package. (An example of an intermediate private data package that allows programmers to access private data via Java interfaces rather than through raw private data bytes is contained in Chapter 3, "Using Telephony Services DEFINITY-Specific Extensions.")

NOTE:
If you are an application programmer who is using JTAPI to develop applications for any switch for which there is a PassageWay Telephony Services driver, ignore this appendix and refer to Chapter 1, "Telephony Services Implementation of JTAPI for All Switches and the DEFINITY Switch." If you want additional TSAPI-specific information about the JTAPI exceptions that are not accessible through standard JTAPI, refer to Chapter 2, "Using Telephony Services Extensions to JTAPI Exceptions."

NOTE:
If you are an application programmer who is using JTAPI to develop applications for the DEFINITY switch, ignore this appendix and refer to Chapter 1, "Telephony Services Implementation of JTAPI for All Switches and the DEFINITY Switch." If you want to take advantage of DEFINITY-specific features that are not accessible through standard JTAPI, refer to Chapter 3, "Telephony Services DEFINITY-Specific Extensions to JTAPI."

Telephony Services Implementation of JTAPI for Private Data

JTAPI's private data mechanism is defined in the java.telephony.privatedata package.

The Lucent Technologies Telephony Services Application Programmer's Interface (TSAPI) implementation adds the ITsapiPeer and ITsapiProvider interfaces to allow an application to set one or more vendors with which it might want to negotiate private data. Applications must invoke the addVendor method on the ITsapiPeer interface before invoking the getProvider method on the interface.

The private data object used is defined as TsapiPrivate. It consists of a vendor name, a byte array of private data, and a tsType value which specifies the escape service to be used.

JTAPI has a different model for private data than TSAPI. If you used private data to program to TSAPI, you have to take the following differences into account to achieve the same result with JTAPI:

- In TSAPI, every request has private data parameters.

  In JTAPI, the associated methods do not have private data parameters. An application must set private data using the setPrivateData method on an object prior to invoking a JTAPI method on that object. The setPrivateData method is defined in the PrivateData interface in the java.telephony.privatedata package. For example, if the desired effect is to send a cstaMakeCall with a private parameter to the switch, the way to achieve that in JTAPI is to first invoke setPrivateData on a Call object and then invoke connect on the Call object.

  Many TSAPI requests have corresponding JTAPI interfaces and methods, as listed in Table A-1. Table A-2 lists the TSAPI requests that do not have corresponding JTAPI interfaces and methods. Therefore, there is no access to the private data for these TSAPI requests.

- In TSAPI, if private data accompanies a confirmation, then it is returned via the acsGetEventBlock or acsGetEventPoll function.

  In JTAPI, there are no confirmation events. An application can get the private data from a confirmation event by using the getPrivateData method on an object after returning from invocation of a method in the object. The getPrivateData method is defined in the PrivateData interface in the java.telephony.privatedata package. For example, if the desired effect is to get the private data from the confirmation, CSTAMakeCallConfEvent; the way to achieve that in JTAPI is to invoke getPrivateData on the Call object after invoking connect on a Call object.

- In TSAPI, if private data accompanies an event, then it is copied via the acsGetEventBlock or acsGetEventPoll function.

  In JTAPI, there are PrivateEvents which are delivered to the observers. The PrivateEvent interface is defined in the java.telephony.privatedata.events package. For example, if the desired effect is to get private data that is associated with cstaDeliveredEvent, the way to achieve that in JTAPI is to extract it from the PrivateEvent that is delivered in an event array to a CallObserver.

  NOTE:
  A cstaDeliveredEvent sets the connection state to ALERTING. If this is a state change, a ConnAlertingEv and PrivateEvent will be in the event array delivered to the CallObserver. If the state was already ALERTING, the PrivateEvent will be in the event array by itself.

  Table A-3 lists TSAPI events and corresponding JTAPI events that might be in the event array in which the PrivateEvent is delivered.

TSAPI Requests with Associated JTAPI Interfaces and Methods

Table A-1 lists the TSAPI requests and associated JTAPI interfaces and methods.

Table A-1. TSAPI Requests with Associated JTAPI Interfaces and Methods

TSAPI Requests	JTAPI Interfaces	JTAPI Methods
cstaMakeCall	Call	connect
cstaClearConnection	Connection	disconnect
acsEnumServerNames	JtapiPeer	getServices
acsOpenStream	JtapiPeer	getProvider
acsCloseStream	Provider	shutdown
cstaAnswerCall	TerminalConnection	answer
cstaSetAgentState	AgentTerminal Agent	addAgent setState
cstaQueryAgentState	Agent	getState
cstaMakePredictiveCall	CallCenterCall	connectPredictive
cstaRouteRegisterReq	RouteAddress	registerRouteCallback
cstaRouteRegisterCancel	RouteAddress	cancelRouteCallback
cstaRouteSelectInv	RouteSession	selectRoute
cstaRouteEndInv	RouteSession	endRoute
cstaSetForwarding	CallControlAddress	setForwarding cancelForwarding
cstaQueryForwarding	CallControlAddress	getForwarding
cstaQueryDoNotDisturb	CallControlAddress	getDoNotDisturb
cstaSetDoNotDisturb	CallControlAddress	setDoNotDisturb
cstaQueryMsgWaitingInd	CallControlAddress	getMessageWaiting
cstaSetMsgWaitingInd	CallControlAddress	setMessageWaiting
cstaClearCall	CallControlCall	drop
cstaConferenceCall	CallControlCall	conference
cstaTransferCall	CallControlCall	transfer
cstaConsultationCall	CallControlCall	consult
cstaDeflectCall	CallControlConnection	redirect
cstaQueryDoNotDisturb	CallControlTerminal	getDoNotDisturb
cstaSetDnd	CallControlTerminal	setDoNotDisturb
cstaPickupCall	CallControlTerminal	pickup
cstaGroupPickupCall	CallControlTerminal	pickupFromGroup
cstaHoldCall	CallControlTerminalConnection	hold
cstaRetrieveCall	CallControlTerminalConnection	unhold
cstaSendPrivateEvent	Private Data	sendPrivateData

TSAPI Requests without Associated JTAPI Mapping

Table A-2 lists the TSAPI requests that do not have corresponding JTAPI interfaces and methods. Therefore, there is no access to the private data for these TSAPI requests.

Table A-2. TSAPI Requests Without Associated JTAPI Mapping

TSAPI Requests
Call Control Services
cstaAlternateCall
cstaCallCompletion
cstaReconnectCall
Supplementary Services
cstaQueryLastNumber
cstaQueryDeviceInfo
Monitor Services
cstaChangeMonitorFilter
FeatureEventReport
CSTACallInfoEvent
Escape Services
cstaEscapeServiceConf
CSTA_ESCAPE_SVC_REQ
Maintenance Services
cstaSysStatReq
cstaSysStatStart
cstaSysStatStop
cstaChangeSysStatFilter
cstaSysStatReqConf
cstaSysStatEvent

Mapping of Possible JTAPI Events to TSAPI Events

Table A-3 lists TSAPI events and the associated possible JTAPI events that might be in the event array in which the PrivateEvent is delivered.

NOTE:
A cstaDeliveredEvent sets the connection state to ALERTING. If this is a state change, a ConnAlertingEv and PrivateEvent will be in the event array delivered to the CallObserver. If the state was already ALERTING, the PrivateEvent will be in the event array by itself.

Table A-3. Mapping of Possible JTAPI Events to TSAPI Events

TSAPI Events	Possible JTAPI Event in Array with Private Event
CSTACallClearedEvent	CallInvalidEv
CSTAMonitorEndedEvent	CallObservationEndedEv
CSTADeliveredEvent	ConnAlertingEv
CSTAEstablishedEvent	ConnConnectedEv
CSTAConnectionClearedEvent	ConnDisconnectedEv
CSTAFailedEvent	ConnFailedEv
CSTADoNotDisturbEvent	CallCtlAddrDoNotDisturbEv
CSTAForwardingEvent	CallCtlAddrForwardEv
CSTAMessageWaitingEvent	CallCtlAddrMessageWaitingEv
CSTAServiceInitiatedEvent	CallCtlConnInitiatedEv
CSTANetworkReachedEvent	CallCtlConnNetworkReachedEv
CSTAQueuedEvent	CallCtlConnQueuedEv
CSTALoggedOffEvent	ACDAddrLoggedOffEv AgentTermLoggedOffEv
CSTALoggedOnEvent	ACDAddrLoggedOnEv AgentTermLoggedOnEv
CSTANotReadyEvent	ACDAddrNotReadyEv AgentTermNotReadyEv
CSTAReadyEvent	ACDAddrReadyEv AgentTermReadyEv
CSTAWorkNotReadyEvent	ACDAddrWorkNotReadyEv AgentTermWorkNotReadyEv
CSTAWorkReadyEvent	ACDAddrWorkReadyEv AgentTermWorkReadyEv
CSTARouteRequestExtEvent	RouteEvent
CSTAReRouteRequestEvent	ReRouteEvent
CSTARouteUsedExtEvent	RouteUsedEvent
CSTARouteEndEvent	RouteEndEvent
CSTARouteRegisterAbortEvent	RouteCallbackEndedEvent

We'd like your opinion ...

Lucent Technologies welcomes your feedback on this document.
Your comments can be of great value in helping us improve our documentation.

PassageWay® Telephony Services
for Windows NT®
Java™ Telephony API (JTAPI)
Client Programmer's Guide
Issue 1.0, October 1997

1. Please rate the effectiveness of this document in the following areas:

	Excellent	Good	Fair	Poor
Ease of Finding Information	❑	❑	❑	❑
Clarity	❑	❑	❑	❑
Completeness	❑	❑	❑	❑
Accuracy	❑	❑	❑	❑
Organization	❑	❑	❑	❑
Appearance	❑	❑	❑	❑
Examples	❑	❑	❑	❑
Overall Satisfaction	❑	❑	❑	❑

2. Please check the ways you feel we could improve this document:

❑ Improve the overview/introduction	❑ Make it more concise
❑ Improve the table of contents	❑ Add more step-by-step procedures/tutorials
❑ Improve the organization	❑ Add more troubleshooting information
❑ Add more figures	❑ Make it less technical
❑ Add more examples	❑ Add more/better quick reference aids
❑ Add more details	❑ Improve the index

Please add details about your major concerns._____
_____
_____

3. What did you like most about this document? _____
_____
_____

4. Feel free to write any comments below or on an attached sheet. _____
_____
_____

If we may contact you concerning your comments, please complete the following:

Name: _____ Telephone Number: (_____)_____

Company/Organization _____ Date: _____

Address: _____

You may FAX your response to (732) 957-4562. Thank you.