

**Lucent Technologies**  
Bell Labs Innovations



# **CentreVu<sup>®</sup> Computer-Telephony for Windows NT<sup>®</sup>**

Release 3.30, Version 2

# **Java<sup>™</sup> Telephony API (JTAPI) Client Programmer's Guide**

Issue 1.1 November 1998



**Copyright © 1998 Lucent Technologies Inc.  
All Rights Reserved  
Printed in U.S.A.**

## **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 number: customers **1 800 451 2100**.

## **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.

CentreVu 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 GLS Information Development, Lucent Technologies, Holmdel, NJ 07733-3030.



---

## Main Contents

This section is intended for those readers who are using Adobe Acrobat to view the document. *If you are reading from a printed version, you can ignore this section.*

There are two ways to navigate through the sections of this file:

- Click on any of the bookmarks visible in the left margin.
- Click on the chapter or section titles in the following table.

<b>Chapter or Section Title</b>	<b>Pages</b>
<a href="#">Title</a>	<a href="#">1 – 2</a>
<a href="#">Copyright</a>	<a href="#">3 – 4</a>
<a href="#">Main Contents (this page)</a>	<a href="#">5 – 6</a>
<a href="#">About This Guide</a>	<a href="#">7 – 14</a>
<a href="#">1 – Telephony Services Implementation of JTAPI for All Switches and the DEFINITY Switch</a>	<a href="#">15 – 44</a>
<a href="#">2 – Using Telephony Services Extensions to JTAPI</a>	<a href="#">45 – 48</a>
<a href="#">Extensions to JTAPI (Interface Index and Exception Index)</a>	<a href="#">49 – 68</a>
<a href="#">3 – Using Telephony Services DEFINITY-Specific Extensions to JTAPI</a>	<a href="#">69 – 74</a>
<a href="#">DEFINITY-Specific Extensions (Interface Index and Class Index)</a>	<a href="#">75 – 156</a>
<a href="#">4 – Using Telephony Services Private Data Extensions to JTAPI</a>	<a href="#">157 – 162</a>
<a href="#">Private Data Extensions (Interface Index and Class Index)</a>	<a href="#">163 – 184</a>
<a href="#">A – Telephony Services Implementation of JTAPI for Private Data</a>	<a href="#">185 – 192</a>
<a href="#">Comment Card</a>	<a href="#">193 – 194</a>



---

# About This Guide

---

## Contents

What is JTAPI?	vii
What is the CentreVu Telephony Services Java Client?	viii
Purpose and Scope	ix
Finding What You Need (Generic JTAPI and DEFINITY-Specific)	x
Finding What You Need (Non-DEFINITY Private Data)	xi
Navigating through the Document	xii
Related Documents	xii





---

## 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 reusable 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 CentreVu 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 CentreVu 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 CentreVu Telephony Services. This implementation provides a programming environment that may be used with any switch for which there is a CentreVu Telephony Services driver.
- the Lucent Technologies CentreVu Telephony Services<sup>1</sup> implementation of JTAPI that provides Telephony Services extensions to JTAPI 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 CentreVu Telephony Services driver.
- the Telephony Services implementation of JTAPI that applies to clients using the DEFINITY switch and the associated CentreVu 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.

---

<sup>1</sup> Hereafter, the phrase “Telephony Services implementation of JTAPI” indicates the Lucent Technologies CentreVu 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 CentreVu 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):
<p><b>Who:</b> an application programmer</p> <p><b>What:</b> Telephony Services generic implementation of JTAPI</p> <p><b>To do:</b> programming with JTAPI for any switch for which there is a CentreVu Telephony Services driver</p> <p><b>Be familiar with:</b></p> <ul style="list-style-type: none"> <li>■ Java</li> <li>■ JTAPI</li> <li>■ TSAPI (optional)</li> </ul>	<p>1) A programming environment for applications used with any switch for which there is a CentreVu Telephony Services driver;</p> <p>2) A programming environment that makes available TSAPI-specific information (optional)</p>	<ul style="list-style-type: none"> <li>■ Chapter 1, "Telephony Services Implementation of JTAPI for All Switches and the DEFINITY Switch"</li> <li>■ Chapter 2, "Using Telephony Services Extensions to JTAPI" (optional)</li> </ul>
<p><b>Who:</b> an application programmer</p> <p><b>What:</b> Telephony Services implementation of JTAPI</p> <p><b>To do:</b> programming with JTAPI for applications used with the DEFINITY switch and the G3PD</p> <p><b>Be familiar with:</b></p> <ul style="list-style-type: none"> <li>■ Java</li> <li>■ JTAPI</li> <li>■ TSAPI (optional)</li> </ul>	<p>1) A programming environment for applications used with the DEFINITY switch and the G3PD;</p> <p>2) A programming environment that makes available DEFINITY-specific features;</p> <p>3) A programming environment that makes available TSAPI-specific information (optional)</p>	<ul style="list-style-type: none"> <li>■ Chapter 1, "Telephony Services Implementation of JTAPI for All Switches and the DEFINITY Switch"</li> <li>■ Chapter 3, "Using Telephony Services DEFINITY-Specific Extensions to JTAPI"</li> <li>■ Chapter 2, "Using Telephony Services Extensions to JTAPI" (optional)</li> </ul>

## 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):
<p><b>Who:</b> An independent switch vendor</p> <p><b>What:</b> Telephony Services implementation of JTAPI for applications using private data</p> <p><b>To do:</b> Programming with JTAPI and TSAPI to produce private data packages for applications used with non-DEFINITY switches and their associated drivers</p> <p><b>Be familiar with:</b></p> <ul style="list-style-type: none"> <li>■ Java</li> <li>■ JTAPI</li> <li>■ TSAPI</li> <li>■ TSAPI's private data mechanism</li> </ul>	<p>A programming environment to produce private data packages that may be used with any non-DEFINITY switch and its associated CentreVu Telephony Services driver</p>	<ul style="list-style-type: none"> <li>■ <a href="#">Chapter 4, "Using Telephony Services Private Data Extensions to JTAPI"</a></li> <li>■ <a href="#">Appendix A, "Telephony Services Implementation of JTAPI for Private Data"</a></li> </ul>
<p><b>Who:</b> An application programmer</p> <p><b>What:</b> Telephony Services implementation of JTAPI for applications using private data</p> <p><b>To do:</b> Programming with JTAPI and TSAPI to use or interpret private data for applications used with non-DEFINITY switches and their associated drivers</p> <p><b>Be familiar with:</b></p> <ul style="list-style-type: none"> <li>■ Java</li> <li>■ JTAPI</li> <li>■ TSAPI</li> <li>■ TSAPI's private data mechanism</li> </ul>	<p>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 CentreVu Telephony Services driver</p>	<ul style="list-style-type: none"> <li>■ <a href="#">Chapter 4, "Using Telephony Services Private Data Extensions to JTAPI"</a></li> <li>■ <a href="#">Appendix A, "Telephony Services Implementation of JTAPI for Private Data"</a></li> </ul>

## **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 will notice a 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 (JTAPI) Programmer's Reference (JTAPI v1.2)*

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 presents the JTAPI v1.2 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.

*CentreVu Computer-Telephony for Windows NT*  
*DEFINITY Enterprise Communications Server 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

<b>Telephony Services Implementation of JTAPI for All Switches and the DEFINITY Switch</b>	<b>1-1</b>
■ Support for JTAPI Core Package	1-2
Implementation Notes	1-4
■ Support for JTAPI Call Center Package	1-5
Implementation Notes	1-6
■ Support for JTAPI Call Center Capabilities Package	1-8
■ Support for JTAPI Call Center Events Package	1-9
Implementation Notes	1-10
■ Support for JTAPI Call Control Package	1-11
Implementation Notes	1-12
■ Support for JTAPI Call Control Capabilities Package	1-14
■ Support for JTAPI Call Control Events Package	1-16
Implementation Notes	1-16
■ Support for JTAPI Capabilities Package	1-17
■ Support for JTAPI Events Package	1-18
■ Support for JTAPI Media Package	1-19
■ Support for JTAPI Media Capabilities Package	1-20
■ Support for JTAPI Media Events Package	1-21
Implementation Notes	1-21
■ Support for JTAPI Phone Package	1-22
■ Support for JTAPI Phone Capabilities Package	1-23
■ Support for JTAPI Phone Events Package	1-24
■ Support for JTAPI Private Data Package	1-25
Implementation Notes	1-25
■ Support for JTAPI Private Data Capabilities Package	1-26
■ Support for JTAPI Private Data Events Package	1-27





---

# 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 CentreVu 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 2](#)) or a method (e.g., [getServices 3](#)), 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
<b>Address</b>	√	√
getName	√	√
getProvider	√	√
getTerminals	√	√
getConnections	√	√
addObserver	√	√
getObservers	√	√
removeObserver	√	√
addCallObserver	√	√
getCallObservers	√	√
removeCallObserver	√	√
getAddressCapabilities	√	√
<b>AddressObserver</b>	√	√
addressChangedEvent	√	√
<b>Call</b>	√	√
getConnections	√	√
getProvider	√	√
getState	√	√
connect	√	√
addObserver	√	√
getObservers	√	√
removeObserver	√	√
getCallCapabilities	√	√
<b>CallObserver</b>	√	√
callChangedEvent	√	√

<b>JTAPI Interfaces and Methods</b>	<b>Supported for All Switches</b>	<b>Supported for the DEFINITY Switch</b>
<b>Connection</b>	√	√
getState	√	√
getCall	√	√
getAddress	√	√
getTerminalConnections	√	√
disconnect <a href="#">1</a>	√	√
getConnectionCapabilities	√	√
<b>JtapiPeer <a href="#">2</a></b>	√	√
getName	√	√
getServices <a href="#">3</a>	√	√
getProvider <a href="#">4</a>	√	√
<b>Provider</b>	√	√
getState	√	√
getName	√	√
getCalls	√	√
getAddress	√	√
getAddresses	√	√
getTerminals	√	√
getTerminal	√	√
shutdown	√	√
createCall	√	√
addObserver	√	√
getObservers	√	√
removeObserver	√	√
getProviderCapabilities	√	√
getCallCapabilities	√	√
getConnectionCapabilities	√	√
getAddressCapabilities	√	√
getTerminalConnectionCapabilities	√	√
getTerminalCapabilities	√	√
<b>ProviderObserver</b>	√	√
providerChangedEvent	√	√
<b>Terminal</b>	√	√
getName	√	√
getProvider	√	√
getAddresses	√	√
getTerminalConnections	√	√
addObserver	√	√
getObservers	√	√
removeObserver	√	√

<b>JTAPI Interfaces and Methods</b>	<b>Supported for All Switches</b>	<b>Supported for the DEFINITY Switch</b>
addCallObserver	√	√
getCallObservers	√	√
removeCallObserver	√	√
getTerminalCapabilities	√	√
<b>TerminalConnection</b>	√	√
getState	√	√
getTerminal	√	√
getConnection	√	√
answer	√	√
getTerminalConnectionCapabilities	√	√
<b>TerminalObserver</b>	√	√
terminalChangedEvent	√	√

### Implementation Notes

- 1 For the DEFINITY switch, the **Connection/disconnect** method must be called with **Connection** in the CONNECTED state. (For all other switches, it can be called with **Connection** in the CONNECTED, ALERTING, INPROGRESS, or FAILED state.)
- 2 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)
```

- 3 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.
- 4 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
<b>ACDAddress</b>	√	√
getLoggedOnAgents <a href="#">1</a>	√	√
getNumberQueued <a href="#">2</a>	√	√
getOldestCallQueued		
getRelativeQueueLoad		
getQueueWaitTime		
getACDManagerAddress		
<b>ACDAddressObserver</b>	√	√
<b>ACDConnection</b>		√
getACDManagerConnection		√
<b>ACDManagerAddress</b>		√
getACDAddresses		
<b>ACDManagerConnection</b>		√
getACDConnections		√
<b>AgentObject</b>	√	√
setState	√	√
getState	√	√
getAgentID <a href="#">3</a>	√	√
getACDAddress	√	√
getAgentAddress	√	√
getAgentTerminal	√	√
<b>AgentTerminal</b>	√	√
addAgent	√	√
removeAgent	√	√
getAgents	√	√
<b>AgentTerminalObserver</b> <a href="#">4</a>	√	√
<b>CallCenterAddress</b>	√	√
addCallObserver	√	√
<b>CallCenterCall</b>	√	√

JTAPI Interfaces and Methods	Supported for All Switches	Supported for the DEFINITY Switch
connectPredictive <a href="#">5</a>	√	√
setApplicationData		
getApplicationData		
getTrunks	√	√
<b>CallCenterCallObserver</b>	√	√
<b>CallCenterProvider</b>	√	√
getRouteableAddresses	√	√
getACDAddresses		√
getACDManagerAddresses		√
<b>CallCenterTrunk</b>	√	√
getName	√	√
getState	√	√
getType	√	√
getCall	√	√
<b>RouteAddress</b>	√	√
registerRouteCallback <a href="#">6</a>	√	√
cancelRouteCallback	√	√
getRouteCallback	√	√
getActiveRouteSessions	√	√
<b>RouteCallback</b>	√	√
routeEvent	√	√
reRouteEvent	√	√
routeUsedEvent	√	√
routeEndEvent	√	√
routeCallbackEndedEvent	√	√
<b>RouteSession</b>	√	√
getRouteAddress	√	√
selectRoute <a href="#">7</a>	√	√
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.

## Implementation Notes (Continued)

- 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.
- 3 The **AgentObject**/`getAgentID` method returns a null String for the DEFINITY switch.
- 4 The **AgentTerminalObserver** only supports the `AgentTermLoggedOnEv` and `AgentTermLoggedOffEv` when the state change is produced through the JTAPI application. In order to monitor agent activity (e.g., agents logging on and off manually), an **ACDAddressObserver** should be added to the **ACDAddress**.
- 5 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.

- 6 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.
- 7 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
<b>ACDAddressCapabilities</b>	√	√
<code>canGetLoggedOnAgents</code>	√	√
<code>canGetNumberQueued</code>	√	√
<code>canGetOldestCallQueued</code>	√	√
<code>canGetRelativeQueueLoad</code>	√	√
<code>canGetQueueWaitTime</code>	√	√
<code>canGetACDManagerAddress</code>	√	√
<b>ACDConnectionCapabilities</b>	√	√
<code>canGetACDManagerConnection</code>	√	√
<b>ACDManagerAddressCapabilities</b>	√	√
<code>canGetACDAddresses</code>	√	√
<b>ACDManagerConnectionCapabilities</b>	√	√
<code>canGetACDConnections</code>	√	√
<b>AgentTerminalCapabilities</b>	√	√
<code>canHandleAgents</code>	√	√
<b>CallCenterAddressCapabilities</b>	√	√
<code>canAddCallObserver</code>	√	√
<b>CallCenterCallCapabilities</b>	√	√
<code>canConnectPredictive</code>	√	√
<code>canHandleApplicationData</code>	√	√
<code>canGetTrunks</code>	√	√
<b>CallCenterProviderCapabilities</b>	√	√
<code>canGetRouteableAddresses</code>	√	√
<code>canGetACDAddresses</code>	√	√
<code>canGetACDManagerAddresses</code>	√	√
<code>canGetTrunks</code>	√	√
<b>RouteAddressCapabilities</b>	√	√
<code>canRouteCalls</code>	√	√



## 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
<b>ACDAddrBusyEv</b>		√
<b>ACDAddrLoggedOffEv</b>	√	√
<b>ACDAddrLoggedOnEv</b>	√	√
<b>ACDAddrNotReadyEv <a href="#">1</a></b>	√	√
<b>ACDAddrReadyEv <a href="#">1</a></b>	√	√
<b>ACDAddrUnknownEv</b>	√	√
<b>ACDAddrWorkNotReadyEv <a href="#">1</a></b>	√	√
<b>ACDAddrWorkReadyEv <a href="#">1</a></b>	√	√
<b>AgentTermBusyEv</b>		√
<b>AgentTermLoggedOffEv</b>	√	√
<b>AgentTermLoggedOnEv</b>	√	√
<b>AgentTermNotReadyEv <a href="#">1</a></b>	√	√
<b>AgentTermReadyEv <a href="#">1</a></b>	√	√
<b>AgentTermUnknownEv</b>	√	√
<b>AgentTermWorkNotReadyEv <a href="#">1</a></b>	√	√
<b>AgentTermWorkReadyEv <a href="#">1</a></b>	√	√
<b>CallCentCallAppDataEv</b>		
<b>RouteCallbackEndedEvent</b>	√	√
<b>RouteEndEvent</b>	√	√
<b>RouteEvent</b>	√	√
<b>RouteSessionEvent</b>	√	√
<b>RouteUsedEvent</b>	√	√

## 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
<b>CallControlAddress</b>	√	√
setForwarding <a href="#">1</a>	√	√
getForwarding	√	√
cancelForwarding	√	√
getDoNotDisturb <a href="#">2</a>	√	√
setDoNotDisturb <a href="#">2</a>	√	√
getMessageWaiting	√	√
setMessageWaiting	√	√
<b>CallControlAddressObserver</b>	√	√
<b>CallControlCall</b>	√	√
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)		
<b>CallControlCallObserver</b>	√	√

<b>JTAPI Interfaces and Methods</b>	<b>Supported for All Switches</b>	<b>Supported for the DEFINITY Switch</b>
<b>CallControlConnection</b>	√	√
getCallControlState	√	√
accept		
reject		
redirect	√	√
addToAddress		
park		
<b>CallControlTerminal</b>	√	√
getDoNotDisturb <a href="#">2</a>	√	√
setDoNotDisturb <a href="#">2</a>	√	√
pickup (Connection pickConnection, Address terminalAddress)	√	√
pickup (TerminalConnection pickTermConn, Address terminalAddress)	√	√
pickup (Address pickAddress, Address terminalAddress)	√	√
pickupFromGroup(String pickupGroup, Address terminalAddress)		
pickupFromGroup(Address terminalAddress)	√	√
<b>CallControlTerminalConnection</b>	√	√
getCallControlState	√	√
hold	√	√
unhold	√	√
join		√
leave		√
<b>CallControlTerminalObserver</b>	√	√

### Implementation Notes

- 1 Our implementation supports the FORWARD\_UNCONDITIONALLY forwarding type only when used in combination with the ALL\_CALLS filter type. When talking to a DEFINITY switch, the only values supported are the FORWARD\_UNCONDITIONALLY forwarding type and the ALL\_CALLS 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
<b>CallControlAddressCapabilities</b>	√	√
canSetForwarding	√	√
canGetForwarding	√	√
canCancelForwarding	√	√
canGetDoNotDisturb	√	√
canSetDoNotDisturb	√	√
canGetMessageWaiting	√	√
canSetMessageWaiting	√	√
<b>CallControlCallCapabilities</b>	√	√
canDrop	√	√
canOffHook	√	√
canSetConferenceController	√	√
canSetTransferController	√	√
canSetTransferEnable	√	√
canSetConferenceEnable	√	√
canTransfer	√	√
canConference	√	√
canAddParty	√	√
canConsult	√	√
<b>CallControlConnectionCapabilities</b>	√	√
canRedirect	√	√
canAddToAddress	√	√
canAccept	√	√
canReject	√	√
canPark	√	√
<b>CallControlTerminalCapabilities</b>	√	√
canGetDoNotDisturb	√	√
canSetDoNotDisturb	√	√
canPickup	√	√

<b>JTAPI Interfaces and Methods</b>	<b>Supported for All Switches</b>	<b>Supported for the DEFINITY Switch</b>
canPickupFromGroup	√	√
<b>CallControlTerminalConnectionCapabilities</b>	√	√
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
<b>CallCtlAddrDoNotDisturbEv</b> <sup>1</sup>	√	√
<b>CallCtlAddrForwardEv</b>	√	√
<b>CallCtlAddrMessageWaitingEv</b>	√	√
<b>CallCtlConnAlertingEv</b>	√	√
<b>CallCtlConnDialingEv</b>		
<b>CallCtlConnDisconnectedEv</b>	√	√
<b>CallCtlConnEstablishedEv</b>	√	√
<b>CallCtlConnFailedEv</b>	√	√
<b>CallCtlConnInitiatedEv</b>	√	√
<b>CallCtlConnNetworkAlertingEv</b>	√	√
<b>CallCtlConnNetworkReachedEv</b>	√	√
<b>CallCtlConnOfferedEv</b>		
<b>CallCtlConnQueuedEv</b>	√	√
<b>CallCtlConnUnknownEv</b>	√	√
<b>CallCtlTermConnBridgedEv</b>	√	√
<b>CallCtlTermConnDroppedEv</b>	√	√
<b>CallCtlTermConnHeldEv</b>	√	√
<b>CallCtlTermConnInUseEv</b>		
<b>CallCtlTermConnRingingEv</b>	√	√
<b>CallCtlTermConnTalkingEv</b>	√	√
<b>CallCtlTermConnUnknownEv</b>	√	√

### Implementation Notes

- 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**

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

## 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
<b>AddrObservationEndedEv</b>	√	√
<b>CallActiveEv</b>	√	√
<b>CallInvalidEv</b>	√	√
<b>CallObservationEndedEv</b>	√	√
<b>ConnAlertingEv</b>	√	√
<b>ConnConnectedEv</b>	√	√
<b>ConnCreatedEv</b>	√	√
<b>ConnDisconnectedEv</b>	√	√
<b>ConnFailedEv</b>	√	√
<b>ConnInProgressEv</b>	√	√
<b>ConnUnknownEv</b>	√	√
<b>ProvInServiceEv</b>	√	√
<b>ProvObservationEndedEv</b>	√	√
<b>ProvOutOfServiceEv</b>	√	√
<b>ProvShutdownEv</b>	√	√
<b>TermConnActiveEv</b>	√	√
<b>TermConnCreatedEv</b>	√	√
<b>TermConnDroppedEv</b>	√	√
<b>TermConnPassiveEv</b>	√	√
<b>TermConnRingingEv</b>	√	√
<b>TermConnUnknownEv</b>	√	√
<b>TermObservationEndedEv</b>	√	√

### **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**

<b>JTAPI Interfaces and Methods</b>	<b>Supported for All Switches</b>	<b>Supported for the DEFINITY Switch</b>
<b>MediaCallObserver</b>		√
<b>MediaTerminalConnection</b>		√
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**

<b>JTAPI Interfaces and Methods</b>	<b>Supported for All Switches</b>	<b>Supported for the DEFINITY Switch</b>
<b>MediaTerminalConnectionCapabilities</b>	√	√
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., **MediaTermConnAvailableEv**, **MediaTermConnDtmfEv**, **MediaTermConnStateEv**, 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
<b>MediaTermConnAvailableEv</b>		
<b>MediaTermConnDtmfEv</b> <a href="#">1</a>		√
<b>MediaTermConnStateEv</b>		
<b>MediaTermConnUnavailableEv</b>		

### 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
<b>PrivateData</b>	√	
<code>getPrivateData</code>	√	
<code>setPrivateData</code> <a href="#">1</a>	√	
<code>sendPrivateData</code> <a href="#">1</a>	√	

### 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
<b>PrivateDataCapabilities</b>	√	√
<code>canSetPrivateData</code>	√	√
<code>canGetPrivateData</code>	√	√
<code>canSendPrivateData</code>	√	√

### **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**

---

<b>JTAPI Interfaces</b>	<b>Supported for All Switches</b>	<b>Supported for the DEFINITY Switch</b>
<b>PrivateAddrEv</b>	√	
<b>PrivateCallEv</b>	√	
<b>PrivateProvEv</b>	√	
<b>PrivateTermEv</b>	√	



---

# Using Telephony Services Extensions to JTAPI

# 2

---

## Contents

<b>Using Telephony Services Extensions to JTAPI</b>	<b>2-1</b>
■ Who Should Be Using These Extensions?	2-1
■ What are the Extensions?	2-2
Extensions to JTAPI Exceptions	2-2
Extensions to JTAPI Provider Events	2-2



---

# Using Telephony Services Extensions to JTAPI

# 2

---

## Using Telephony Services Extensions to JTAPI

---

**NOTE:**

This chapter describes non-standard additions to JTAPI. This package is available only from the CentreVu 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 that can be used to program applications for any switch for which there is a CentreVu 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 to develop applications which will be used with any switch for which there is a CentreVu 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?

---

There are two types of extensions: extensions to JTAPI exceptions and extensions to JTAPI Provider events.

### Extensions to JTAPI Exceptions

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*.

### Extensions to JTAPI Provider Events

Telephony Services defines additional JTAPI Provider events. These events provide more detailed Provider state changes. These TSAPI Provider states map to JTAPI Provider states as follows:

**Table 2-1. Mapping of TSAPI to JTAPI Provider States**

---

TSAPI Provider State	JTAPI Provider State
ITsapiProvider.TSAPI_OUT_OF_SERVICE	Provider.OUT_OF_SERVICE
ITsapiProvider.TSAPI_INITIALIZING	Provider.OUT_OF_SERVICE
ITsapiProvider.TSAPI_IN_SERVICE	Provider.IN_SERVICE
ITsapiProvider.TSAPI_SHUTDOWN	Provider.SHUTDOWN



---

## **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 Troubleshooting section of the Telephony Services Administration and Maintenance document (netmangd.pdf).

---

### Variable Index

- o [ACS](#)  
Error Type of ACS.
- o [CSTA](#)  
Error Type of CSTA.
- o [EC INVALID CONF](#)  
Error Code implying confirmation is invalid.
- o [EC NORMAL](#)  
Error Code of NORMAL.
- o [EC PROVIDER OUT OF SERVICE](#)  
Error Code implying Provider is OUT\_OF\_SERVICE.
- o [INTERNAL](#)  
Failure is internal to this Jtapi implementation.
- o [JTAPI](#)  
Failed to meet some Jtapi condition.
- o [NORMAL](#)  
Error Type of Normal.

### Method Index

- o [getErrorCode\(\)](#)  
Returns the error code.
- o [getErrorType\(\)](#)  
Returns the error type.

### Variables

- o **NORMAL**

```
public static final int NORMAL
```

Error Type of Normal.

**o ACS**

```
public static final int ACS
```

Error Type of ACS.

**o CSTA**

```
public static final int CSTA
```

Error Type of CSTA.

**o JTAPI**

```
public static final int JTAPI
```

Failed to meet some Jtapi condition.

**o INTERNAL**

```
public static final int INTERNAL
```

Failure is internal to this Jtapi implementation.

**o EC\_NORMAL**

```
public static final int EC_NORMAL
```

Error Code of NORMAL.

**o EC\_INVALID\_CONF**

```
public static final int EC_INVALID_CONF
```

Error Code implying confirmation is invalid.

**o 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

- o [getErrorCode\(\)](#)  
Returns the error code.
- o [getErrorType\(\)](#)  
Returns the error type.

### 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

- o [getErrorCode\(\)](#)  
Returns the error code.
- o [getErrorType\(\)](#)  
Returns the error type.

### 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

- o [getErrorCode\(\)](#)  
Returns the error code.
- o [getErrorType\(\)](#)  
Returns the error type.

### 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

- o [getErrorCode\(\)](#)  
Returns the error code.
- o [getErrorType\(\)](#)  
Returns the error type.

## 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

- o [getErrorCode\(\)](#)  
Returns the error code.
- o [getErrorType\(\)](#)  
Returns the error type.

### 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

- o [getErrorCode\(\)](#)  
Returns the error code.
- o [getErrorType\(\)](#)  
Returns the error type.

## 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

- o [getErrorCode\(\)](#)  
Returns the error code.
- o [getErrorType\(\)](#)  
Returns the error type.

## Methods

### o getErrorType

```
public int getErrorType()
```

Returns the error type.

## **o 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

- o [getErrorCode\(\)](#)  
Returns the error code.
- o [getErrorType\(\)](#)  
Returns the error type.

## 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

<b>Using Telephony Services DEFINITY-Specific Extensions</b>	3-1
■ Who Should Be Using These Extensions?	3-1
■ How Should the Extensions be Used?	3-2



---

# 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 CentreVu 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 CentreVu 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 CentreVu 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 that is not accessible through standard JTAPI, refer to [Chapter 2](#), "Using Telephony Services 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."

**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.

**Table 3-1. DEFINITY Features Provided by DEFINITY-Specific Extensions to JTAPI**

---

Feature Name and Description	Class or Interface	Returned/Used by Methods in Class or Interface
<b>Advice of Charge</b> Reports network charges incurred by outgoing trunk calls (supported by G3V5)	LucentChargeAdviceEvent	LucentV5Provider
<b>Agent Work Mode</b> Specifies the overriding mode of the Agent; affects the cycle of the possibly occurring Agents states. G3V5 adds support for: reason code.		LucentAgent LucentAgentStateInfo LucentTerminal LucentV5Terminal LucentV5AgentStateInfo
<b>Call Classifier Information</b> Provides information on call classifier port usage	CallClassifierInfo	LucentProvider
<b>Collect Digits</b> Allows a route request to wait for a specified number of digits to be collected		LucentRouteSession



<b>Feature Name and Description</b>	<b>Class or Interface</b>	<b>Returned/Used by Methods in Class or Interface</b>
<b>Dial-Ahead Digits</b> Allows a route request to place digits in a dial-ahead buffer		LucentRouteSession
<b>Direct Agent Calls</b> Allows calls to be made to and from specific logged-in ACD Agents		LucentCall LucentRouteSession
<b>Dropping Resources</b> Allows specific switch resources to be dropped from the call		LucentConnection LucentTerminalConnection
<b>Flexible Billing</b> Allows changing the billing rate for incoming 900-type calls (supported by G3V5)		LucentV5Call
<b>Integrated Directory Name</b> Allows the G3 Integrated Directory Database name to be returned		LucentAddress LucentTerminal
<b>Look-Ahead Interflow Information</b> May be used by a routing server application to determine the proper destination of a call	LookaheadInfo	LucentCallInfo OriginalCallInfo
<b>Lucent Call Information</b> 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. G3V5 adds support for: Universal Call ID, Originator Type, and Flex Billing Flag.	LucentCallInfo LucentV5CallInfo	Implemented by Lucent call objects, route session objects, and CallControlCall events.
<b>Message Waiting Application Information</b> Indicates which types of applications have enabled message waiting		LucentAddress LucentAddressMsgWaitingEvent
<b>Network Progress Information</b> Contains supplementary call progress information from the ISDN Progress Indicator Information Element G3V5 adds support for: trunk.	NetworkProgressInfo V5NetworkProgressInfo	LucentConnNetworkReachedEvent

<b>Feature Name and Description</b>	<b>Class or Interface</b>	<b>Returned/Used by Methods in Class or Interface</b>
<b>Original Call Information</b> Contains information about the original call in conjunction with the Call.consult() service. G3V5 adds support for: Universal Call ID, Originator Type, and Flex Billing Flag.	OriginalCallInfo V5OriginalCallInfo	LucentCallInfo LucentV5CallInfo
<b>Priority Calls</b> Enables priority calling		LucentCall LucentRouteSession
<b>Selective Listen</b> Allows control of listen paths between parties on a conference call (supported by G3V5)		LucentV5TerminalConnection
<b>Single Step Conference</b> Adds another party to a call (added party does not alert; used mainly for service observing) (supported by G3V5)		LucentV5Call
<b>Supervisor Assist Calls</b> Allows logged-in ACD Agents to place calls to a supervisor's extension		LucentCall
<b>Switch Date and Time Information</b> Returns the current date and time from the switch		LucentProvider
<b>Trunk Group Information</b> Provides information on trunk group usage	TrunkGroupInfo	LucentProvider
<b>Universal Call ID</b> A call identifier that is globally unique across switches and the public network (supported by G3V5)		LucentV5CallInfo (extended by LucentV5Call)
<b>User Entered Code</b> 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
<b>User-to-User Information</b> 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

- [ITsapiAddress](#)
- [ITsapiAddressMsgWaitingEvent](#)
- [ITsapiAgent](#)
- [ITsapiCall](#)
- [ITsapiCallInfo](#)
- [ITsapiConnNetworkReachedEvent](#)
- [ITsapiConnection](#)
- [ITsapiRouteSession](#)
- [ITsapiTerminal](#)
- [ITsapiTerminalConnection](#)
- [LucentAddress](#)
- [LucentAddressMsgWaitingEvent](#)
- [LucentAgent](#)
- [LucentCall](#)
- [LucentCallInfo](#)
- [LucentConnNetworkReachedEvent](#)
- [LucentConnection](#)
- [LucentProvider](#)
- [LucentRouteSession](#)
- [LucentTerminal](#)
- [LucentTerminalConnection](#)
- [LucentV5Call](#)
- [LucentV5CallInfo](#)
- [LucentV5Connection](#)
- [LucentV5Provider](#)
- [LucentV5Terminal](#)
- [LucentV5TerminalConnection](#)

### Class Index

- [CallClassifierInfo](#)
- [LookaheadInfo](#)
- [LucentAgentStateInfo](#)
- [LucentBillType](#)
- [LucentChargeAdviceEvent](#)
- [LucentChargeError](#)
- [LucentChargeType](#)
- [LucentV5AgentStateInfo](#)
- [NetworkProgressInfo](#)
- [OriginalCallInfo](#)
- [TrunkGroupInfo](#)

- [UserEnteredCode](#)
- [UserToUserInfo](#)
- [V5NetworkProgressInfo](#)
- [V5OriginalCallInfo](#)

---

## **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 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 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 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 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 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 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 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` to return the `Call` object associated with the `RouteSession`.

---

### Method Index

o [getCall\(\)](#)

Returns the `Call` object associated with this `RouteSession`.

### Methods

o **getCall**

```
public abstract Call getCall()
```

Returns the `Call` object associated with this `RouteSession`. This `Call` reference remains valid throughout the lifetime of the `RouteSession` object, despite the state of the `RouteSession` object. This `Call` reference does not change once the `RouteSession` object has been created.

**Returns:**

The call object associated with this `RouteSession`.

---

---

## **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 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 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

o [MWI CTI](#)

The message waiting indicator has been enabled via CTI.

o [MWI LWC](#)

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

o [MWI MCS](#)

The message waiting indicator has been enabled via Message Center.

o [MWI PROPMGT](#)

The message waiting indicator has been enabled via Property Management.

o [MWI VOICE](#)

The message waiting indicator has been enabled via Voice Messaging.

### Method Index

o [getDirectoryName\(\)](#)

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

o [getMessageWaitingBits\(\)](#)

Returns a bit-mask indicating which applications have enabled the message waiting indicator at this Address.

### 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.

#### o **MWI\_PROPMGT**

```
public static final int MWI_PROPMGT
```

The message waiting indicator has been enabled via Property Management.

#### o **MWI\_LWC**

```
public static final int MWI_LWC
```

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

#### o **MWI\_CTI**

```
public static final int MWI_CTI
```

The message waiting indicator has been enabled via CTI.

## **Methods**

#### o **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.

#### o **getDirectoryName**

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

Returns the DEFINITY 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

### o [getMessageWaitingBits\(\)](#)

Returns a bit-mask indicating which applications have enabled the message waiting indicator at this Address.

## 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

o [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](#)

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 .

o [MODE\\_NONE](#)

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

### Method Index

o [getStateInfo\(\)](#)

This returns this Agent's state and workMode.

o [setState\(int, int\)](#)

This method overrides `Agent.setState()` to add the Lucent-specific parameter `workMode`.

### 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, TsapiInvalidStateException
```

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 UII in the ISDN SETUP message over the PRI trunk to establish the call. The local and the remote switch include the UII in the alerting, connected, disconnected and route request events.

**See Also:**  
[UserToUserInfo](#)

---

## Method Index

- o [connect](#)(`LucentTerminal`, `LucentAddress`, `String`, `boolean`, `UserToUserInfo`)  
Similar to the standard `connect()`, with the addition of Lucent-specific call parameters.
- o [connectDirectAgent](#)(`LucentTerminal`, `LucentAddress`, `LucentAgent`, `boolean`, `UserToUserInfo`)  
Places a direct call to a specific logged-in ACD agent.
- o [connectPredictive](#)(`LucentTerminal`, `LucentAddress`, `String`, `int`, `int`, `int`, `int`, `boolean`, `UserToUserInfo`)  
Similar to the standard `connectPredictive()`, with the addition of Lucent-specific call parameters.
- o [connectSupervisorAssist](#)(`LucentAgent`, `String`, `UserToUserInfo`)  
Places a call from a logged-in ACD agent to a supervisor's extension.

- o [consult](#)([LucentTerminalConnection](#), [String](#), [boolean](#), [UserToUserInfo](#))  
Similar to the standard `consult()`, with the addition of Lucent-specific call parameters.
- o [consultDirectAgent](#)([LucentTerminalConnection](#), [LucentAgent](#), [boolean](#), [UserToUserInfo](#))  
Places a consultation call with a specific logged-in ACD agent.
- o [consultSupervisorAssist](#)([LucentTerminalConnection](#), [ACDAddress](#), [String](#), [UserToUserInfo](#))  
Places a consultation call from a logged-in ACD agent to a supervisor's extension.

## Methods

### o connect

```
public abstract Connection[] connect(LucentTerminal origterm,
                                     LucentAddress origaddr,
                                     String dialedDigits,
                                     boolean priorityCall,
                                     UserToUserInfo userInfo) throws TsapiResourceUnavailableException;
```

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](#)

### o connectDirectAgent

```
public abstract Connection[] connectDirectAgent(LucentTerminal origterm,
                                                LucentAddress origaddr,
                                                LucentAgent calledAgent,
                                                boolean priorityCall,
                                                UserToUserInfo userInfo) throws TsapiResourceUnavailableException;
```

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(LucentAgent callingAgent,  
String dialedDigits,  
UserToUserInfo userInfo) throws TsapiResourceUnavail
```

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(LucentTerminal originatorTerminal,  
LucentAddress origAddress,  
String dialedDigits,  
int connectionState,  
int maxRings,  
int answeringTreatment,  
int answeringEndpointType,  
boolean priorityCall,  
UserToUserInfo userInfo) throws TsapiResourceUnavail
```

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(LucentTerminalConnection termconn,  
                                     String address,  
                                     boolean priorityCall,  
                                     UserToUserInfo userInfo) throws TsapiInvalidStateException, TsapiException
```

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(LucentTerminalConnection termconn,  
                                                LucentAgent calledAgent,  
                                                boolean priorityCall,  
                                                UserToUserInfo userInfo) throws TsapiInvalidStateException, TsapiException
```

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(LucentTerminalConnection termconn,  
                                                    ACDAddress split,  
                                                    String address,  
                                                    UserToUserInfo userInfo) throws TsapiInvalidStateException, TsapiException
```

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, the route session object, and on certain 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

- o [AR ANSWER MACHINE DETECTED](#)
- o [AR ANSWER NORMAL](#)
- o [AR ANSWER TIMED](#)
- o [AR ANSWER VOICE ENERGY](#)
- o [AR IN QUEUE](#)
- o [AR NONE](#)
- o [AR SIT INEFFECTIVE OTHER](#)
- o [AR SIT INTERCEPT](#)
- o [AR SIT NO CIRCUIT](#)
- o [AR SIT REORDER](#)
- o [AR SIT UNKNOWN](#)
- o [AR SIT VACANT CODE](#)

### Method Index

- o [getDeliveringACDAddress\(\)](#)  
For a connection to an `AgentTerminal`, `getDeliveringACDAddress` returns the `ACDAddress` that this call was delivered through to the `AgentTerminal`.
- o [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`.
- o [getLookaheadInfo\(\)](#)  
Returns lookahead interflow information associated with the call event.
- o [getOriginalCallInfo\(\)](#)  
Returns original call information associated with the call event.

- o **[getReason\(\)](#)**  
Specifies the reason for the last event sent for Connections and TerminalConnections on the Call or the Call.
- o **[getTrunk\(\)](#)**  
Returns the trunk from which the call originated.
- o **[getUserEnteredCode\(\)](#)**  
Returns call prompting digits associated with the call event.
- o **[getUserToUserInfo\(\)](#)**  
Returns user-to-user information associated with the call event.

## Variables

### o **AR\_NONE**

```
public static final short AR_NONE
```

### o **AR\_ANSWER\_NORMAL**

```
public static final short AR_ANSWER_NORMAL
```

### o **AR\_ANSWER\_TIMED**

```
public static final short AR_ANSWER_TIMED
```

### o **AR\_ANSWER\_VOICE\_ENERGY**

```
public static final short AR_ANSWER_VOICE_ENERGY
```

### o **AR\_ANSWER\_MACHINE\_DETECTED**

```
public static final short AR_ANSWER_MACHINE_DETECTED
```

### o **AR\_SIT\_REORDER**

```
public static final short AR_SIT_REORDER
```

### o **AR\_SIT\_NO\_CIRCUIT**

```
public static final short AR_SIT_NO_CIRCUIT
```

### o **AR\_SIT\_INTERCEPT**

```
public static final short AR_SIT_INTERCEPT
```

### o **AR\_SIT\_VACANT\_CODE**

```
public static final short AR_SIT_VACANT_CODE
```

### o **AR\_SIT\_INEFFECTIVE\_OTHER**

```
public static final short AR_SIT_INEFFECTIVE_OTHER
```

#### o **AR\_SIT\_UNKNOWN**

```
public static final short AR_SIT_UNKNOWN
```

#### o **AR\_IN\_QUEUE**

```
public static final short AR_IN_QUEUE
```

## Methods

#### o **getUserToUserInfo**

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

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

#### o **getLookaheadInfo**

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

Returns lookahead interflow information associated with the call event.

#### o **getUserEnteredCode**

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

Returns call prompting digits associated with the call event.

#### o **getOriginalCallInfo**

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

Returns original call information associated with the call event.

#### o **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`.

#### o **getDeliveringACDAddress**

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

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

**o getTrunk**

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

Returns the trunk from which the call originated.

**o 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

o [getNetworkProgressInfo\(\)](#)  
Get the supplementary call progress information

## 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

- o [DR\\_CALL\\_CLASSIFIER](#)  
Drop a call classifier from the call.
- o [DR\\_NONE](#)
- o [DR\\_TONE\\_GENERATOR](#)  
Drop a tone generator from the call.

### Method Index

- o [disconnect](#)(short, UserToUserInfo)  
Similar to the standard `disconnect()`, with the addition of Lucent-specific parameters.

### 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, TsapiException
```

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

- o [getCallClassifierInfo\(\)](#)  
Returns information on call classifier port usage.
- o [getSwitchDateAndTime\(\)](#)  
Returns current date and time from the switch.
- o [getTrunkGroupInfo\(String\)](#)  
Returns trunk usage information on the specified trunk group.

### Methods

#### o **getTrunkGroupInfo**

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

Returns trunk usage information on the specified trunk group.

#### o **getCallClassifierInfo**

```
public abstract CallClassifierInfo getCallClassifierInfo() throws TsapiMethodNotSupportedException
```

Returns information on call classifier port usage.

#### o **getSwitchDateAndTime**

```
public abstract Date getSwitchDateAndTime() throws TsapiMethodNotSupportedException
```

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.

The route session object which implements this interface also implements the `LucentCallInfo` interface.

---

### Method Index

- o [selectRoute](#)(String, boolean, UserToUserInfo)  
Similar to the standard `selectRoute()`, with the addition of Lucent-specific call parameters.
- o [selectRouteAndCollect](#)(String, int, int, boolean, UserToUserInfo)  
Routes a call and requests DTMF digit collection.
- o [selectRouteDirectAgent](#)(LucentAgent, boolean, UserToUserInfo)  
Routes a direct agent call to a specific logged-in ACD agent.
- o [selectRouteWithDigits](#)(String, String, boolean, UserToUserInfo)  
Routes a call and places digits in a dial-ahead digit buffer.

### 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(LucentAgent 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 TsapiMethodNotSupportedE
```

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 TsapiMethodNotSupportedE
```

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

- o [addAgent](#)(`LucentAddress`, `ACDAddress`, `int`, `int`, `String`, `String`)  
This method overrides `Terminal.addAgent()` to add the Lucent-specific parameter `workMode`.
- o [getDirectoryName](#)()  
Return Directory name of this Terminal.

### Methods

#### o `addAgent`

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

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 (may be `null`).

`initialState` – specifies the initial state of the agent. Valid states are `Agent.READY`, `Agent.NOT_READY` and `Agent.LOG_IN`.

`workMode` – specifies the work mode this Agent should be set to. Valid workModes are `LucentAgent.MODE_AUTO_IN` and

`LucentAgent.MODE_MANUAL_IN`.

`agentID` – is the Agent's ID.

`password` – is the Agent's 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

- o [DR\\_CALL\\_CLASSIFIER](#)  
Drop a call classifier from the call..
- o [DR\\_NONE](#)
- o [DR\\_TONE\\_GENERATOR](#)  
Drop a tone generator from the call..

### Method Index

- o [leave](#)(short, UserToUserInfo)  
Similar to the standard `leave()`, with the addition of Lucent-specific parameters.

### 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, TsapiMethodN
```

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](#)

---



---

## Interface `com.lucent.jtapi.tsapi.LucentV5Call`

public interface **LucentV5Call**

extends [ITsapiCall](#), [LucentCall](#), [LucentV5CallInfo](#)

The `LucentV5Call` interface extends `ITsapiCall` with Lucent-specific features. When a Provider is bound to a Lucent DEFINITY switch with PBX Driver Version 5 private data, this interface may be used to access additional capabilities.

---

### Method Index

o [addParty](#)(String, boolean)

Adds a new party to an active Call, without alerting at the added party (intended mainly for service observing).

o [setBillRate](#)(short, float)

This service supports the AT&T MultiQuest 900 Vari-A-Bill Service to change the rate for an incoming 900-type call.

### Methods

o **addParty**

```
public abstract Connection addParty(String newParty,  
                                   boolean isActive) throws TsapiInvalidStateException, TsapiInval
```

Adds a new party to an active Call, without alerting at the added party (intended mainly for service observing). If *isActive* is *false*, the added party will have its talk path disabled. This "Single-Step Conference" feature is specific to DEFINITY G3V6.

**Parameters:**

*newParty* – The telephone address of the party to be added.

*isActive* – Specifies whether the party is added in active or silent mode.

**Returns:**

The new Connection associated with the added party.

o **setBillRate**

```
public abstract void setBillRate(short billType,  
                                 float billRate) throws TsapiInvalidArgumentException, TsapiMethodN
```

This service supports the AT&T MultiQuest 900 Vari-A-Bill Service to change the rate for an incoming 900-type call. The client application can request this service at any time after the call has been answered and before the call is cleared.

**Parameters:**

billType – Specifies the rate treatment for the call. See LucentBillType for allowed values.

billRate – Specifies the rate according to the treatment indicated by billType. If BT\_FREE\_CALL is specified, billRate is ignored. This is a floating point number. The rate should not be less than zero, and a maximum is set for each 900-number as part of the provisioning process (in the 4E switch).

**See Also:**

[LucentBillType](#)

---

---

## Interface `com.lucent.jtapi.tsapi.LucentV5CallInfo`

public interface **LucentV5CallInfo**  
extends [LucentCallInfo](#)

The `LucentV5CallInfo` interface provides access to call information from Lucent DEFINITY switches with PBX Driver Version 5 private data. These methods are implemented on the call object, the route session object, and on certain call control call events. For example, if a `CallControlCallObserver` receives a `CallCtlConnAlertingEv`, it may be cast to `LucentV5CallInfo` to use the `getUCID()` method. These methods may return null if the requested data is not available.

---

### Method Index

- o [canSetBillRate\(\)](#)  
Returns the Flexible Billing flag, which indicates whether the `setBillRate()` method is valid for this call
- o [getCallOriginatorType\(\)](#)  
Get the originator type for this call, such as coin call, 800 service call, or cellular call.
- o [getUCID\(\)](#)  
Get the Universal Call ID for this call.
- o [hasCallOriginatorType\(\)](#)  
Query whether `CallOriginatorType` is available for this call.

### Methods

#### o `getUCID`

```
public abstract String getUCID()
```

Get the Universal Call ID for this call. (This feature requires DEFINITY G3V6).

#### o `getCallOriginatorType`

```
public abstract int getCallOriginatorType()
```

Get the originator type for this call, such as coin call, 800 service call, or cellular call. This information is from the network, not from the DEFINITY switch. The type is defined in the Bell Communications Research (Bellcore) publication, "Local

Exchange Routing Guide," (document number TR-EOP-000085). A list of defined codes, as of June 1994, follows:

00	Identified line - no special treatment
01	Multiparty - ANI cannot be provided
02	ANI failure
06	Hotel/Motel - DN not accompanied by automatic room ID
07	Special operator handling required
20	AIOD - Listed DN of PBX sent
23	Coin or Non-Coin - line status unknown
24	800 Service Call
27	Coin Call
29	Prison/Inmate Service
30-32	Intercept
34	Telco Operator Handled Call
40-49	Locally determined by carrier
52	Out WATS
60	Telecommunication Relay Service (TRS) - Station Paid
61	Type 1 Cellular
62	Type 2 Cellular
63	Romer Cellular
66	TRS - From Hotel/Motel
67	TRS - From restricted line
70	pay station
93	Virtual Network call

### **o hasCallOriginatorType**

```
public abstract boolean hasCallOriginatorType()
```

Query whether CallOriginatorType is available for this call.

### **o canSetBillRate**

```
public abstract boolean canSetBillRate()
```

Returns the Flexible Billing flag, which indicates whether the setBillRate() method is valid for this call

---

---

## Interface `com.lucent.jtapi.tsapi.LucentV5Connection`

public interface **LucentV5Connection**  
extends [LucentConnection](#)

The `LucentV5Connection` interface extends `LucentConnection` with features specific to DEFINITY G3 PBX Driver Version 5 private data. When a Provider is bound to a DEFINITY switch which supports V5 private data, this interface may be used to access additional capabilities.

The Selective Listening service allows an application to prevent a specific party on a call from hearing anything said by another specific party (or all other parties) on the call. It allows an application to put a non-bridged Connection's listening path on listen-hold with respect to a selected `TerminalConnection` or non-bridged Connection (`partyToHold`), or to all other parties. The selected party(s) may be stations or trunks. A party that has been listen-held may continue to talk and be heard by other connected parties on the call since this service does not affect the talking or listening path of any other party. A party will be able to hear parties on the call from which it has not been listen-held, but will not be able to hear any party from which it has been listen-held. This service will also allow the listen-held party to be unheld (i.e., to again hear the other party(s) on the call).

The Selective Listening service is also available on `LucentV5TerminalConnection`.

---

### Method Index

- o [listenHold](#)(`LucentConnection`)  
Places a non-bridged Connection's listening path on listen-hold with respect to the specified non-bridged Connection.
- o [listenHold](#)(`LucentTerminalConnection`)  
Places a non-bridged Connection's listening path on listen-hold with respect to the specified `TerminalConnection`.
- o [listenUnhold](#)(`LucentConnection`)  
Takes a non-bridged Connection's listening path off listen-hold with respect to the specified non-bridged Connection.
- o [listenUnhold](#)(`LucentTerminalConnection`)  
Takes a non-bridged Connection's listening path off listen-hold with respect to the specified `TerminalConnection`.

## Methods

### o listenHold

```
public abstract void listenHold(LucentTerminalConnection partyToHold) throws TsapiInvalidStateException
```

Places a non-bridged Connection's listening path on listen-hold with respect to the specified TerminalConnection. If partyToHold is null, the operation applies to all other parties on the call. If the Connection has multiple TerminalConnections, a TsapiInvalidArgumentException is thrown; in this case, use a TerminalConnection instead.

### o listenUnhold

```
public abstract void listenUnhold(LucentTerminalConnection partyToUnhold) throws TsapiInvalidStateException
```

Takes a non-bridged Connection's listening path off listen-hold with respect to the specified TerminalConnection. If partyToUnhold is null, the operation applies to all other parties on the call. If the Connection has multiple TerminalConnections, a TsapiInvalidArgumentException is thrown; in this case, use a TerminalConnection instead.

### o listenHold

```
public abstract void listenHold(LucentConnection partyToHold) throws TsapiInvalidStateException, TsapiInvalidArgumentEx
```

Places a non-bridged Connection's listening path on listen-hold with respect to the specified non-bridged Connection. If partyToHold is null, the operation applies to all other parties on the call. If either Connection has multiple TerminalConnections, a TsapiInvalidArgumentException is thrown; in this case, use a TerminalConnection instead.

### o listenUnhold

```
public abstract void listenUnhold(LucentConnection partyToUnhold) throws TsapiInvalidStateException
```

Takes a non-bridged Connection's listening path off listen-hold with respect to the specified non-bridged Connection. If partyToUnhold is null, the operation applies to all other parties on the call. If either Connection has multiple TerminalConnections, a TsapiInvalidArgumentException is thrown; in this case, use a TerminalConnection instead.

---

---

## Interface `com.lucent.jtapi.tsapi.LucentV5Provider`

public interface **LucentV5Provider**  
extends [LucentProvider](#)

LucentV5Provider adds the Advice Of Charge feature.

---

### Method Index

- o [setAdviceOfCharge](#)(boolean)  
Activate or deactivate the Advice Of Charge feature.

### Methods

- o **setAdviceOfCharge**

public abstract void setAdviceOfCharge(boolean flag) throws [TsapiMethodNotSupportedException](#)

Activate or deactivate the Advice Of Charge feature. Setting the flag to true will enable Charge Advice Events.

---

---

## Interface `com.lucent.jtapi.tsapi.LucentV5Terminal`

public interface **LucentV5Terminal**  
extends [LucentTerminal](#)

The `LucentV5Terminal` interface extends the `LucentTerminal` interface.

---

### Method Index

- o [addAgent](#)(`LucentAddress`, `ACDAddress`, `int`, `int`, `int`, `String`, `String`)  
This method overrides `Terminal.addAgent()` to add the Lucent DEFINITY G3 PBX Driver Version 5 private data-specific parameter *reasonCode*.

### Methods

#### o `addAgent`

```
public abstract Agent addAgent(LucentAddress agentAddress,  
                               ACDAddress acdAddress,  
                               int initialState,  
                               int workMode,  
                               int reasonCode,  
                               String agentID,  
                               String password) throws TsapiInvalidArgumentException, TsapiInvalidS
```

This method overrides `Terminal.addAgent()` to add the Lucent DEFINITY G3 PBX Driver Version 5 private data-specific parameter *reasonCode*. 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 (may be null).

initialState – specifies the initial state of the agent. Valid states are Agent.READY, Agent.NOT\_READY and Agent.LOG\_IN.

workMode – specifies the work mode this Agent should be set to. Valid workModes are LucentAgent.MODE\_AUTO\_IN and LucentAgent.MODE\_MANUAL\_IN.

reasonCode – Application-defined reason code (1-9).

agentID – is the Agent's ID.

password – is the Agent's 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.

---

---

## Interface

### **com.lucent.jtapi.tsapi.LucentV5TerminalConnection**

public interface **LucentV5TerminalConnection**

extends [LucentTerminalConnection](#)

The **LucentV5TerminalConnection** interface extends **LucentTerminalConnection** with features specific to DEFINITY G3 PBX Driver Version 5 private data. When a Provider is bound to a DEFINITY switch which supports V5 private data, this interface may be used to access additional capabilities.

The Selective Listening service allows an application to prevent a specific party on a call from hearing anything said by another specific party (or all other parties) on the call. It allows an application to put a **TerminalConnection**'s listening path on listen–hold with respect to a selected **TerminalConnection** or non–bridged **Connection** (`partyToHold`), or to all other parties. The selected party(s) may be stations or trunks. A party that has been listen–held may continue to talk and be heard by other connected parties on the call since this service does not affect the talking or listening path of any other party. A party will be able to hear parties on the call from which it has not been listen–held, but will not be able to hear any party from which it has been listen–held. This service will also allow the listen–held party to be unheld (i.e., to again hear the other party(s) on the call).

The Selective Listening service is also available on **LucentV5Connection**.

---

## Method Index

- o [listenHold](#)(**LucentConnection**)  
Places a **TerminalConnection**'s listening path on listen–hold with respect to the specified non–bridged **Connection**.
- o [listenHold](#)(**LucentTerminalConnection**)  
Places a **TerminalConnection**'s listening path on listen–hold with respect to the specified **TerminalConnection**.
- o [listenUnhold](#)(**LucentConnection**)  
Takes a **TerminalConnection**'s listening path off listen–hold with respect to the specified non–bridged **Connection**.
- o [listenUnhold](#)(**LucentTerminalConnection**)  
Takes a **TerminalConnection**'s listening path off listen–hold with respect to the specified **TerminalConnection**.

## Methods

### o listenHold

```
public abstract void listenHold(LucentTerminalConnection partyToHold) throws TsapiInvalidStateException
```

Places a TerminalConnection's listening path on listen–hold with respect to the specified TerminalConnection. If partyToHold is null, the operation applies to all other parties on the call.

### o listenUnhold

```
public abstract void listenUnhold(LucentTerminalConnection partyToUnhold) throws TsapiInvalidStateException
```

Takes a TerminalConnection's listening path off listen–hold with respect to the specified TerminalConnection. If partyToUnhold is null, the operation applies to all other parties on the call.

### o listenHold

```
public abstract void listenHold(LucentConnection partyToHold) throws TsapiInvalidStateException, TsapiInvalidArgumentException
```

Places a TerminalConnection's listening path on listen–hold with respect to the specified non–bridged Connection. If partyToHold is null, the operation applies to all other parties on the call. If the Connection has multiple TerminalConnections, a TsapiInvalidArgumentException is thrown; in this case, specify a TerminalConnection instead.

### o listenUnhold

```
public abstract void listenUnhold(LucentConnection partyToUnhold) throws TsapiInvalidStateException, TsapiInvalidArgumentException
```

Takes a TerminalConnection's listening path off listen–hold with respect to the specified non–bridged Connection. If partyToUnhold is null, the operation applies to all other parties on the call. If the Connection has multiple TerminalConnections, a TsapiInvalidArgumentException is thrown; in this case, specify a TerminalConnection instead.

---

---

## Class `com.lucent.jtapi.tsapi.CallClassifierInfo`

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

---

public final class **CallClassifierInfo**

Provides information on call classifier port usage.

---

### Variable Index

o [\*\*numAvailPorts\*\*](#)

The number of available call classifier ports.

o [\*\*numInUsePorts\*\*](#)

The number of in-use call classifier ports.

### 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.LookaheadInfo
```

---

public class **LookaheadInfo**

Lookahead interflow is a DEFINITY 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.

This information, when available, is obtained via the `LucentCallInfo.getLookaheadInfo()` method.

### See Also:

[LucentCallInfo](#)

---

## Variable Index

- o [LAI ALL INTERFLOW](#)
- o [LAI HIGH](#)
- o [LAI LOW](#)
- o [LAI MEDIUM](#)
- o [LAI NOT IN QUEUE](#)
- o [LAI THRESHOLD INTERFLOW](#)
- o [LAI TOP](#)
- o [LAI VECTORING INTERFLOW](#)

## Method Index

- o [getHours\(\)](#)  
Gets the 'hours' part of the event timestamp.
- o [getMinutes\(\)](#)  
Gets the 'minutes' part of the event timestamp.

- o [getPriority\(\)](#)  
Priority of the interflowed call.
- o [getSeconds\(\)](#)  
Gets the 'seconds' part of the event timestamp.
- o [getSourceVDN\(\)](#)  
Returns the address of the VDN which overflowed the call.
- o [getType\(\)](#)  
Type of interflow.

## Variables

### o LAI\_ALL\_INTERFLOW

```
public static final short LAI_ALL_INTERFLOW
```

### o LAI\_THRESHOLD\_INTERFLOW

```
public static final short LAI_THRESHOLD_INTERFLOW
```

### o LAI\_VECTORING\_INTERFLOW

```
public static final short LAI_VECTORING_INTERFLOW
```

### o LAI\_NOT\_IN\_QUEUE

```
public static final short LAI_NOT_IN_QUEUE
```

### o LAI\_LOW

```
public static final short LAI_LOW
```

### o LAI\_MEDIUM

```
public static final short LAI_MEDIUM
```

### o LAI\_HIGH

```
public static final short LAI_HIGH
```

### o LAI\_TOP

```
public static final short LAI_TOP
```

## Methods

### o getType

```
public short getType()
```

Type of interflow. Possible values are LAI\_ALL\_INTERFLOW, LAI\_THRESHOLD\_INTERFLOW, and LAI\_VECTORING\_INTERFLOW.

#### **o 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.

#### **o getHours**

```
public int getHours()
```

Gets the 'hours' part of the event timestamp.

#### **o getMinutes**

```
public int getMinutes()
```

Gets the 'minutes' part of the event timestamp.

#### **o getSeconds**

```
public int getSeconds()
```

Gets the 'seconds' part of the event timestamp.

#### **o 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 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 Also:

[LucentAgent](#)

---

## Variable Index

### o [state](#)

State of Agent.

### o [workMode](#)

Work Mode for Agent.

## Variables

### o **state**

```
public int state
```

State of Agent.

### o **workMode**

```
public int workMode
```

Work Mode for Agent.

---





---

## Class com.lucent.jtapi.tsapi.LucentBillType

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

---

public final class **LucentBillType**

This class defines constants used with the LucentV5Call.setBillRate() method.

**See Also:**

[LucentV5Call](#)

---

### Variable Index

- o [BT\\_FLAT\\_RATE](#)  
time independent
- o [BT\\_FREE\\_CALL](#)  
no charge
- o [BT\\_NEW\\_RATE](#)  
new rate
- o [BT\\_PREMIUM\\_CHARGE](#)  
a flat charge in addition to the existing rate
- o [BT\\_PREMIUM\\_CREDIT](#)  
a flat negative charge in addition to the existing rate

### Variables

#### o **BT\_NEW\_RATE**

```
public static final short BT_NEW_RATE
```

new rate

#### o **BT\_FLAT\_RATE**

```
public static final short BT_FLAT_RATE
```

time independent

**o BT\_PREMIUM\_CHARGE**

public static final short BT\_PREMIUM\_CHARGE

a flat charge in addition to the existing rate

**o BT\_PREMIUM\_CREDIT**

public static final short BT\_PREMIUM\_CREDIT

a flat negative charge in addition to the existing rate

**o BT\_FREE\_CALL**

public static final short BT\_FREE\_CALL

no charge

---

---

## Class `com.lucent.jtapi.tsapi.LucentChargeAdviceEvent`

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

---

```
public final class LucentChargeAdviceEvent
```

---

### Method Index

- o [getCall\(\)](#)  
The call for which this Charge Advice event is being reported
- o [getCalledAddress\(\)](#)  
The external address which was dialed
- o [getCharge\(\)](#)  
The number of units charged
- o [getChargeError\(\)](#)  
Charge-related error.
- o [getChargeType\(\)](#)  
The type of charge being reported
- o [getChargingAddress\(\)](#)  
The address being charged for this call
- o [getTrunk\(\)](#)  
The trunk reporting the charge

### Methods

#### o **getCall**

```
public final LucentCall getCall()
```

The call for which this Charge Advice event is being reported

#### o **getCalledAddress**

```
public final LucentAddress getCalledAddress()
```

The external address which was dialed

### o **getChargingAddress**

```
public final LucentAddress getChargingAddress()
```

The address being charged for this call

### o **getTrunk**

```
public final CallCenterTrunk getTrunk()
```

The trunk reporting the charge

### o **getCharge**

```
public final int getCharge()
```

The number of units charged

### o **getChargeType**

```
public final short getChargeType()
```

The type of charge being reported

#### **See Also:**

[LucentChargeType](#)

### o **getChargeError**

```
public final short getChargeError()
```

Charge-related error.

#### **See Also:**

[LucentChargeError](#)

---

---

## Class com.lucent.jtapi.tsapi.LucentChargeError

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

---

public final class **LucentChargeError**

This class defines constants used with the `LucentChargeAdviceEvent.getChargeError()` method.

**See Also:**

[LucentChargeAdviceEvent](#)

---

### Variable Index

- o [CE\\_CHARGE\\_TOO\\_LARGE](#)  
Charge provide by the network is too large
- o [CE\\_LESS\\_FINAL\\_CHARGE](#)  
Final charge provide by the network is less than a previous charge
- o [CE\\_NETWORK\\_BUSY](#)  
Too many calls are waiting for their final charge from the network
- o [CE\\_NO\\_FINAL\\_CHARGE](#)  
Network failed to provide a final charge for the call
- o [CE\\_NONE](#)  
No error

### Variables

#### o **CE\_NONE**

```
public static final short CE_NONE
```

No error

#### o **CE\_NO\_FINAL\_CHARGE**

```
public static final short CE_NO_FINAL_CHARGE
```

Network failed to provide a final charge for the call

**o CE\_LESS\_FINAL\_CHARGE**

```
public static final short CE_LESS_FINAL_CHARGE
```

Final charge provide by the network is less than a previous charge

**o CE\_CHARGE\_TOO\_LARGE**

```
public static final short CE_CHARGE_TOO_LARGE
```

Charge provide by the network is too large

**o CE\_NETWORK\_BUSY**

```
public static final short CE_NETWORK_BUSY
```

Too many calls are waiting for their final charge from the network

---

---

## Class com.lucent.jtapi.tsapi.LucentChargeType

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

---

public final class **LucentChargeType**

This class defines constants used with the `LucentChargeAdviceEvent.getChargeType()` method.

**See Also:**

[LucentChargeAdviceEvent](#)

---

### Variable Index

o [CT\\_FINAL\\_CHARGE](#)

This charge is sent by the trunk when a call is dropped.

o [CT\\_INTERMEDIATE\\_CHARGE](#)

This is a charge sent by the trunk while the call is active.

o [CT\\_SPLIT\\_CHARGE](#)

CDR outgoing call splitting is used to divide the charge for a call among different users.

### Variables

o **CT\_INTERMEDIATE\_CHARGE**

```
public static final short CT_INTERMEDIATE_CHARGE
```

This is a charge sent by the trunk while the call is active. The charge amounts reported are cumulative. If a call receives two or more consecutive intermediate charges, then the amount from the last intermediate charge replaces the amount(s) of the previous intermediate charges. The amounts are not added to produce a total charge.

o **CT\_FINAL\_CHARGE**



```
public static final short CT_FINAL_CHARGE
```

This charge is sent by the trunk when a call is dropped. If call CDR outgoing call splitting is not enabled, then the final charge reflects the charge for the entire call.

#### o **CT\_SPLIT\_CHARGE**

```
public static final short CT_SPLIT_CHARGE
```

CDR outgoing call splitting is used to divide the charge for a call among different users. For example, if an outgoing call is placed by one station and transferred to a second station, and if CDR call splitting is enabled, then CDR and the Charge Advice Events would charge the first station up to the time of the transfer, and second station after that. A split charge reflects the charge for the call up to the time the split charge is sent (starting at the beginning of the call, or at the previous split charge). Any Charge Advice Event received after a split charge will reflect only that portion of the charge that took place after the split charge. If split charges are received for a call, then the total charge for the call can be computed by adding the split charges and the final charge.

---

---

## Class `com.lucent.jtapi.tsapi.LucentV5AgentStateInfo`

```
java.lang.Object
|
+----<a href="http://www.lucent.com/jtapi/tsapi/LucentAgentStateInfo">com.lucent.jtapi.tsapi.LucentAgentStateInfo
|
+----<a href="http://www.lucent.com/jtapi/tsapi/LucentV5AgentStateInfo">com.lucent.jtapi.tsapi.LucentV5AgentStateInfo
```

---

```
public final class LucentV5AgentStateInfo
extends LucentAgentStateInfo
```

This is the object that is returned by the query `getStateInfo` in `LucentAgent`. It returns the state, `workMode`, and application-defined `reasonCode` for the Agent.

See `LucentAgent` for details.

---

### Variable Index

o [reasonCode](#)  
Application-defined reason code (1-9)

### Variables

o **reasonCode**

```
public int reasonCode
```

Application-defined reason code (1-9)

---

---

## Class com.lucent.jtapi.tsapi.NetworkProgressInfo

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

---

public class **NetworkProgressInfo**

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

---

### Variable Index

o [PD CALL OFF ISDN](#)

o [PD CALL ON ISDN](#)

o [PD DEST NOT ISDN](#)

o [PD INBAND](#)

o [PD ORIG NOT ISDN](#)

o [PL PRIV REMOTE](#)

o [PL PUB LOCAL](#)

o [PL PUB REMOTE](#)

o [PL USER](#)

o [progressDescription](#)

Specifies the progress description in a Progress Indicator Information Element from the PRI network.

o [progressLocation](#)

Specifies the progress location in a Progress Indicator Information Element from the PRI network.

### 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.OriginalCallInfo
```

---

public class **OriginalCallInfo**

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.

This information, when available, is obtained via the LucentCallInfo.getOriginalCallInfo() method.

**See Also:**

[LucentCallInfo](#)

---

### Variable Index

- o [OR CONFERENCED](#)
- o [OR CONSULTATION](#)
- o [OR NEW CALL](#)
- o [OR NONE](#)
- o [OR TRANSFERRED](#)

### Method Index

- o [getCalledDevice\(\)](#)  
Get the original called device for this call.
- o [getCallingDevice\(\)](#)  
Get the original calling device for this call.
- o [getLookaheadInfo\(\)](#)  
Get the original lookahead information for this call.
- o [getReason\(\)](#)  
Get the reason code for this OriginalCallInfo.
- o [getTrunk\(\)](#)

Get the original trunk device for this call.

o [\*\*getUserEnteredCode\(\)\*\*](#)

Get the original collected digits for this call.

o [\*\*getUserToUserInfo\(\)\*\*](#)

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

## Variables

o **OR\_NONE**

```
public static final short OR_NONE
```

o **OR\_CONSULTATION**

```
public static final short OR_CONSULTATION
```

o **OR\_CONFERENCED**

```
public static final short OR_CONFERENCED
```

o **OR\_TRANSFERRED**

```
public static final short OR_TRANSFERRED
```

o **OR\_NEW\_CALL**

```
public static final short OR_NEW_CALL
```

## Methods

o **getReason**

```
public short getReason()
```

Get the reason code for this OriginalCallInfo. Possible values are OR\_NONE, OR\_CONSULTATION, OR\_CONFERENCED, OR\_TRANSFERRED, and OR\_NEW\_CALL.

o **getCallingDevice**

```
public Address getCallingDevice()
```

Get the original calling device for this call.

o **getCalledDevice**

```
public Address getCalledDevice()
```

Get the original called device for this call.

**o getTrunk**

```
public CallCenterTrunk getTrunk()
```

Get the original trunk device for this call.

**o getUserToUserInfo**

```
public UserToUserInfo getUserToUserInfo()
```

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

**o getLookaheadInfo**

```
public LookaheadInfo getLookaheadInfo()
```

Get the original lookahead information for this call.

**o 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.TrunkGroupInfo
```

---

public final class **TrunkGroupInfo**

Provides information on trunk group usage.

---

### Variable Index

o [\*\*idleTrunks\*\*](#)

The number of idle trunks.

o [\*\*usedTrunks\*\*](#)

The number of in-use trunks.

### 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.UserEnteredCode
```

---

### public final class **UserEnteredCode**

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

This information, when available, is obtained via the `LucentCallInfo.getUserEnteredCode()` method.

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.

Note: VDNs are represented through the `ACDManagerAddress` interface.

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, using `CallCenterAddress.addCallObserver` with the `remain` flag set to `true`. 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.

**See Also:**

[LucentCallInfo](#)

---

## Variable Index

- o [UE ANY](#)
- o [UE CALL PROMPTER](#)
- o [UE COLLECT](#)
- o [UE DATA BASE PROVIDED](#)
- o [UE ENTERED](#)
- o [UE LOGIN DIGITS](#)
- o [UE TONE DETECTOR](#)

## Method Index

- o [getCollectVDN\(\)](#)  
Returns the ACDManagerAddress of the VDN which collected the digits
- o [getDigits\(\)](#)  
Returns the collected digits
- o [getIndicator\(\)](#)  
Returns UE\_COLLECT or UE\_ENTERED
- o [getType\(\)](#)  
Returns the type of digits collected

## 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
```

### o **UE\_COLLECT**

```
public static final short UE_COLLECT
```

### o **UE\_ENTERED**

```
public static final short UE_ENTERED
```

## **Methods**

### o **getType**

```
public short getType()
```

Returns the type of digits collected

### o **getIndicator**

```
public short getIndicator()
```

Returns UE\_COLLECT or UE\_ENTERED

### o **getDigits**

```
public String getDigits()
```

Returns the collected digits

### o **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.UserToUserInfo
```

---

### public final class **UserToUserInfo**

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 UII in the ISDN SETUP message over the PRI trunk to establish the call. The local and the remote switch include the UII in the alerting, connected, disconnected and route request events.

This information, when available, is obtained via the `LucentCallInfo.getUserToUserInfo()` method.

#### See Also:

[LucentCallInfo](#)

---

### Constructor Index

- o [UserToUserInfo](#)(byte[])  
construct a UserToUserInfo object from a byte array
- o [UserToUserInfo](#)(String)  
construct a UserToUserInfo object from an ASCII string

### Method Index

- o [getBytes](#)()  
return user-to-user info as a (binary) byte array
- o [getString](#)()  
return user-to-user info as an ASCII string
- o [isAscii](#)()

query whether sender encoded UII as ASCII or binary

## 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 UII as ASCII or binary

---



---

## Class `com.lucent.jtapi.tsapi.V5NetworkProgressInfo`

```
java.lang.Object
|
+----<a href="http://javadoc.sun.com/6.0/docs/api/com/lucent/jtapi/tsapi/NetworkProgressInfo.html">com.lucent.jtapi.tsapi.NetworkProgressInfo
    |
    +----com.lucent.jtapi.tsapi.V5NetworkProgressInfo
```

---

```
public final class V5NetworkProgressInfo
extends NetworkProgressInfo
```

Adds DEFINITY G3V6-specific data to the NetworkProgressInfo event

---

### Variable Index

o [trunk](#)

### Variables

o **trunk**

```
public TsapiTrunk trunk
```

---

---

## Class `com.lucent.jtapi.tsapi.V5OriginalCallInfo`

```
java.lang.Object
|
+----<a href="#">com.lucent.jtapi.tsapi.OriginalCallInfo
|
+----<a href="#">com.lucent.jtapi.tsapi.V5OriginalCallInfo
```

---

public final class **V5OriginalCallInfo**  
extends [OriginalCallInfo](#)

This class adds DEFINITY G3 PBX Driver Version 5 private data extensions to OriginalCallInfo.

---

### Method Index

- o [canSetBillRate\(\)](#)  
Returns the Flexible Billing flag, which indicates whether the setBillRate() method is valid for this call
- o [getCallOriginatorType\(\)](#)  
Get the originator type for this call, such as coin call, 800 service call, or cellular call.
- o [getUCID\(\)](#)  
Get the Universal Call ID for this call.
- o [hasCallOriginatorType\(\)](#)  
Query whether CallOriginatorType is available for this call.

### Methods

#### o [getUCID](#)

```
public String getUCID()
```

Get the Universal Call ID for this call. (Requires DEFINITY G3V6.)

#### o [getCallOriginatorType](#)

```
public int getCallOriginatorType()
```

Get the originator type for this call, such as coin call, 800 service call, or cellular call. This information is from the network, not from the DEFINITY switch. The type is defined in the Bell Communications Research (Bellcore) publication, "Local Exchange Routing Guide," (document number TR-EOP-000085). A list of defined codes, as of June 1994, follows:

00	Identified line - no special treatment
01	Multiparty - ANI cannot be provided
02	ANI failure
06	Hotel/Motel - DN not accompanied by automatic room ID
07	Special operator handling required
20	AIOD - Listed DN of PBX sent
23	Coin or Non-Coin - line status unknown
24	800 Service Call
27	Coin Call
29	Prison/Inmate Service
30-32	Intercept
34	Telco Operator Handled Call
40-49	Locally determined by carrier
52	Out WATS
60	Telecommunication Relay Service (TRS) - Station Paid
61	Type 1 Cellular
62	Type 2 Cellular
63	Romer Cellular
66	TRS - From Hotel/Motel
67	TRS - From restricted line
70	pay station
93	Virtual Network call

### **o hasCallOriginatorType**

```
public boolean hasCallOriginatorType()
```

Query whether CallOriginatorType is available for this call.

### **o canSetBillRate**

```
public boolean canSetBillRate()
```

Returns the Flexible Billing flag, which indicates whether the setBillRate() method is valid for this call

---



---

# Using Telephony Services Private Data Extensions to JTAPI

# 4

---

## Contents

<b>Using Telephony Services Private Data Extensions</b>	4-1
■ Who Should Be Using These Extensions?	4-1
■ How Should the Extensions be Used?	4-2
Initialization of Private Data	4-2
Using TsapiPrivate as a JTAPI Private Data Object	4-3
Converting TSAPI Constructs to JTAPI Objects	4-3
Converting JTAPI Objects to TSAPI Constructs	4-4



---

# 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 CentreVu 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 CentreVu 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 that is not accessible through standard JTAPI, refer to [Chapter 2](#), "Using Telephony Services Extensions to JTAPI."

⇒ **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 response 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.

**Table 4-1. Mapping of TSAPI Constructs to JTAPI Objects**

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.

**Table 4-2. Mapping JTAPI Objects to TSAPI Constructs**

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

### o [getTsapiConnectionID\(\)](#)

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

## 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 data with.

---

### Method Index

o [addVendor](#)(String, String)

This method can be used to set the vendor the application wants to exchange data with.

### 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 data with. The interfaces for 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](#)

---

## Variable Index

### o [TSAPI IN SERVICE](#)

The `ITsapiProvider.TSAPI_IN_SERVICE` state maps to the core JTAPI `Provider.IN_SERVICE` state.

### o [TSAPI INITIALIZING](#)

The `ITsapiProvider.TSAPI_OUT_OF_SERVICE` and `ITsapiProvider.TSAPI_INITIALIZING` states map to the core JTAPI `Provider.OUT_OF_SERVICE` state.

### o [TSAPI OUT OF SERVICE](#)

The `ITsapiProvider.TSAPI_OUT_OF_SERVICE` and `ITsapiProvider.TSAPI_INITIALIZING` states map to the core JTAPI `Provider.OUT_OF_SERVICE` state.

### o [TSAPI SHUTDOWN](#)

The `ITsapiProvider.TSAPI_SHUTDOWN` state maps to the core JTAPI `Provider.SHUTDOWN` state.

## Method Index

### o [getTsapiState\(\)](#)

Returns the TSAPI state of the provider.

### o [getVendor\(\)](#)

Returns the data vendor name.

### o [getVendorVersion\(\)](#)

Returns the negotiated vendor data version.

### o [updateAddresses\(\)](#)

Query the TServer to update the list of Addresses returned by `getAddresses()`

## Variables

### o TSAPI\_OUT\_OF\_SERVICE

```
public static final int TSAPI_OUT_OF_SERVICE
```

The `ITSapiProvider.TSAPI_OUT_OF_SERVICE` and `ITSapiProvider.TSAPI_INITIALIZING` states map to the core `JTAPI Provider.OUT_OF_SERVICE` state.

### o TSAPI\_INITIALIZING

```
public static final int TSAPI_INITIALIZING
```

The `ITSapiProvider.TSAPI_OUT_OF_SERVICE` and `ITSapiProvider.TSAPI_INITIALIZING` states map to the core `JTAPI Provider.OUT_OF_SERVICE` state. The `ITSapiProvider.TSAPI_INITIALIZING` state implies that the provider is available to perform most actions, but hasn't completed its entire initialization. In this state, actions such as `provider.getAddress(String)` and `provider.getTerminal(String)` may succeed when the resulting `Address` or `Terminal` is actually outside of the provider's domain (and, hence, the request should really fail). Other actions, such as `provider.getAddresses()` and `provider.getTerminals()` may be requested in this state but will block until the provider goes `ITSapiProvider.TSAPI_IN_SERVICE`.

### o TSAPI\_IN\_SERVICE

```
public static final int TSAPI_IN_SERVICE
```

The `ITSapiProvider.TSAPI_IN_SERVICE` state maps to the core `JTAPI Provider.IN_SERVICE` state.

### o TSAPI\_SHUTDOWN

```
public static final int TSAPI_SHUTDOWN
```

The `ITSapiProvider.TSAPI_SHUTDOWN` state maps to the core `JTAPI Provider.SHUTDOWN` state.

## Methods

### o getTsapiState

```
public abstract int getTsapiState()
```

Returns the TSAPI state of the provider.

### **o getVendor**

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

Returns the data vendor name.

### **o getVendorVersion**

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

Returns the negotiated vendor data version.

### **o updateAddresses**

```
public abstract void updateAddresses()
```

Query the TServer to update the list of Addresses returned by getAddresses()

---



---

## 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

- o [getAddress](#)(ExtendedDeviceID)  
Returns a JTAPI Address associated with a TSAPI Extended Device ID.
- o [getCall](#)(int)  
Returns a JTAPI Call associated with a TSAPI Call ID.
- o [getConnection](#)(ConnectionID, Address)  
Returns a JTAPI Connection associated with a TSAPI Connection ID and the specified JTAPI Address.
- o [getTerminal](#)(ExtendedDeviceID)  
Returns a JTAPI Terminal associated with a TSAPI Extended Device ID.
- o [getTerminalConnection](#)(ConnectionID, Terminal)  
Returns a JTAPI TerminalConnection associated with a TSAPI Connection ID and the specified JTAPI Terminal.

### 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

o [getRouteCrossRefID\(\)](#)

Retrieves the TSAPI RouteCrossReferenceID associated with a JTAPI Route Session.

o [getRouteRegisterID\(\)](#)

Retrieves the TSAPI RouteRegisterID with a JTAPI Route Session.

### 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.ConnectionID
```

---

public final class **ConnectionID**

---

### Variable Index

- o [DYNAMIC\\_ID](#)
- o [STATIC\\_ID](#)

### Method Index

- o [equals](#)(Object)
- o [hashCode](#)()
- o [toString](#)()

### 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.ExtendedDeviceID
```

---

public final class **ExtendedDeviceID**

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

**See Also:**

[ITsapiProviderPrivate](#)

---

## Variable Index

- o [DEVICE IDENTIFIER](#)  
Device ID Type.
- o [EXPLICIT PRIVATE ABBREVIATED](#)  
Device ID Type.
- o [EXPLICIT PRIVATE LEVEL1 REGIONAL NUMBER](#)  
Device ID Type.
- o [EXPLICIT PRIVATE LEVEL2 REGIONAL NUMBER](#)  
Device ID Type.
- o [EXPLICIT PRIVATE LEVEL3 REGIONAL NUMBER](#)  
Device ID Type.
- o [EXPLICIT PRIVATE LOCAL NUMBER](#)  
Device ID Type.
- o [EXPLICIT PRIVATE PTN SPECIFIC NUMBER](#)  
Device ID Type.
- o [EXPLICIT PRIVATE UNKNOWN](#)  
Device ID Type.
- o [EXPLICIT PUBLIC ABBREVIATED](#)  
Device ID Type.
- o [EXPLICIT PUBLIC INTERNATIONAL](#)  
Device ID Type.

- o [EXPLICIT\\_PUBLIC\\_NATIONAL](#)  
Device ID Type.
- o [EXPLICIT\\_PUBLIC\\_NETWORK\\_SPECIFIC](#)  
Device ID Type.
- o [EXPLICIT\\_PUBLIC\\_SUBSCRIBER](#)  
Device ID Type.
- o [EXPLICIT\\_PUBLIC\\_UNKNOWN](#)  
Device ID Type.
- o [ID\\_NOT\\_KNOWN](#)  
Device ID Status of *ID\_NOT\_KNOWN* indicates the Device ID is not known.
- o [ID\\_NOT\\_REQUIRED](#)  
Device ID Status of *ID\_NOT\_REQUIRED* indicates the Device ID is not required.
- o [ID\\_PROVIDED](#)  
Device ID Status of *ID\_PROVIDED* indicates the Device ID is valid
- o [IMPLICIT\\_PRIVATE](#)  
Device ID Type.
- o [IMPLICIT\\_PUBLIC](#)  
Device ID Type.
- o [OTHER\\_PLAN](#)  
Device ID Type.
- o [TRUNK\\_GROUP\\_IDENTIFIER](#)  
Device ID Type.
- o [TRUNK\\_IDENTIFIER](#)  
Device ID Type.

## Constructor Index

- o [ExtendedDeviceID](#)(String, short, short)  
Construct an ExtendedDeviceID.

## Method Index

- o [toString](#)()

## Variables

### o **DEVICE\_IDENTIFIER**

```
public static final short DEVICE_IDENTIFIER
```

Device ID Type. Ignored if Device ID Status is not *ID\_PROVIDED*

### o **IMPLICIT\_PUBLIC**

```
public static final short IMPLICIT_PUBLIC
```

Device ID Type. Ignored if Device ID Status is not *ID\_PROVIDED*



**o EXPLICIT\_PUBLIC\_UNKNOWN**

```
public static final short EXPLICIT_PUBLIC_UNKNOWN
```

Device ID Type. Ignored if Device ID Status is not *ID\_PROVIDED*

**o EXPLICIT\_PUBLIC\_INTERNATIONAL**

```
public static final short EXPLICIT_PUBLIC_INTERNATIONAL
```

Device ID Type. Ignored if Device ID Status is not *ID\_PROVIDED*

**o EXPLICIT\_PUBLIC\_NATIONAL**

```
public static final short EXPLICIT_PUBLIC_NATIONAL
```

Device ID Type. Ignored if Device ID Status is not *ID\_PROVIDED*

**o EXPLICIT\_PUBLIC\_NETWORK\_SPECIFIC**

```
public static final short EXPLICIT_PUBLIC_NETWORK_SPECIFIC
```

Device ID Type. Ignored if Device ID Status is not *ID\_PROVIDED*

**o EXPLICIT\_PUBLIC\_SUBSCRIBER**

```
public static final short EXPLICIT_PUBLIC_SUBSCRIBER
```

Device ID Type. Ignored if Device ID Status is not *ID\_PROVIDED*

**o EXPLICIT\_PUBLIC\_ABBREVIATED**

```
public static final short EXPLICIT_PUBLIC_ABBREVIATED
```

Device ID Type. Ignored if Device ID Status is not *ID\_PROVIDED*

**o IMPLICIT\_PRIVATE**

```
public static final short IMPLICIT_PRIVATE
```

Device ID Type. Ignored if Device ID Status is not *ID\_PROVIDED*

**o EXPLICIT\_PRIVATE\_UNKNOWN**

```
public static final short EXPLICIT_PRIVATE_UNKNOWN
```

Device ID Type. Ignored if Device ID Status is not *ID\_PROVIDED*

**o 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*

**o 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*

**o 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*

**o 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*

**o EXPLICIT\_PRIVATE\_LOCAL\_NUMBER**

```
public static final short EXPLICIT_PRIVATE_LOCAL_NUMBER
```

Device ID Type. Ignored if Device ID Status is not *ID\_PROVIDED*

**o EXPLICIT\_PRIVATE\_ABBREVIATED**

```
public static final short EXPLICIT_PRIVATE_ABBREVIATED
```

Device ID Type. Ignored if Device ID Status is not *ID\_PROVIDED*

**o OTHER\_PLAN**

```
public static final short OTHER_PLAN
```

Device ID Type. Ignored if Device ID Status is not *ID\_PROVIDED*

**o 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*).

## **Methods**

### o **toString**

```
public String toString()
```

**Overrides:**

[toString](#) in class Object

---

---

## 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 data interfaces. Where JTAPI specifies that a 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 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 data to be used.

### See Also:

[addVendor](#)

---

## Variable Index

- o [data](#)
- o [tsType](#)
- o [vendor](#)

## Constructor Index

- o [TsapiPrivate](#)(byte[])  
Construct a TSAPI data object.
- o [TsapiPrivate](#)(byte[], boolean)  
Construct a TSAPI data object.

## Method Index

### o [getData\(\)](#)

Return the byte array containing the raw 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 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 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 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 data.

---





---

# Telephony Services Implementation of JTAPI for Private Data



---

## Contents

<b>Telephony Services Implementation of JTAPI for Private Data</b>	A-2
■ TSAPI Requests with Associated JTAPI Interfaces and Methods	A-4
■ TSAPI Requests without Associated JTAPI Mapping	A-5
■ Mapping of Possible JTAPI Events to TSAPI Events	A-6



---

# Telephony Services Implementation of JTAPI for Private Data



---

This appendix describes the level of support the CentreVu 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 CentreVu 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 that is not accessible through standard JTAPI, refer to [Chapter 2](#), “Using Telephony Services Extensions to JTAPI.”

**⇒ 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	CallControlTerminalCon nection	hold
cstaRetrieveCall	CallControlTerminalCon nection	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**

---

<b>TSAPI Requests</b>
<b>Call Control Services</b>
cstaAlternateCall
cstaCallCompletion
cstaReconnectCall
<b>Supplementary Services</b>
cstaQueryLastNumber
cstaQueryDeviceInfo
<b>Monitor Services</b>
cstaChangeMonitorFilter
FeatureEventReport
CSTACallInfoEvent
<b>Escape Services</b>
cstaEscapeServiceConf
CSTA_ESCAPE_SVC_REQ
<b>Maintenance Services</b>
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
CSTARReadyEvent	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.

**CentreVu® Computer-Telephony  
for Windows NT®  
Java™ Telephony API (JTAPI)  
Client Programmer's Guide  
Issue 1.1 November 1998**

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

	Excellent	Good	Fair	Poor
Ease of Finding Information .....	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Clarity .....	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Completeness .....	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Accuracy .....	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization .....	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Appearance .....	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Examples .....	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Overall Satisfaction .....	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

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

- |  |   |
|--|---|
| <input type="checkbox"/> Improve the overview/introduction | <input type="checkbox"/> Make it more concise                       |
| <input type="checkbox"/> Improve the table of contents     | <input type="checkbox"/> Add more step-by-step procedures/tutorials |
| <input type="checkbox"/> Improve the organization          | <input type="checkbox"/> Add more troubleshooting information       |
| <input type="checkbox"/> Add more figures                  | <input type="checkbox"/> Make it less technical                     |
| <input type="checkbox"/> Add more examples                 | <input type="checkbox"/> Add more/better quick reference aids       |
| <input type="checkbox"/> Add more details                  | <input type="checkbox"/> 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) 817-4562. Thank you.**

