



**CentreVu Computer Telephony R10.1 V1**

**Java™ Telephony API (JTAPI) for DEFINITY Enterprise  
Communications Server**

**Programmer's Reference**

**Issue 1 — December 2001**

**Copyright 2001 Avaya, 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. Avaya 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. Avaya will not be responsible for any charges that result from such unauthorized use.

### **Avaya 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**.

### **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 Avaya logotype are registered trademarks of Avaya, Inc..

Windows NT is a registered trademark of Microsoft Corp.

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

### **Acknowledgment**

This document was prepared by Avaya University Information Development, Holmdel, NJ 07733-3030.

## About This Document

This document is for Java programmers who are developing CTI based client applications. The information on this document complements the Telephony Services Java Client software SDK on the CentreVu Computer Telephony CD ROM.

## About the Avaya Implementation of JTAPI

- Avaya currently implements JTAPI 1.2, which has a consistent event delivery model across the core and extension packages - in particular, event delivery is done through observers across the core, callcontrol and callcenter packages. (JTAPI 1.2 is the last major ECTF release with enhancements to callcontrol and callcenter packages.)
- Avaya chose not to implement support for the JTAPI 1.3 API since it does not address the needs of CTI and Call Center developers. JTAPI 1.3, as a phased release of the Listener pattern, does not carry the Listener changes from the core package over into the callcontrol and callcenter extension packages. (JTAPI 1.3 is the current generally available ECTF release of JTAPI. It adds the mobility package, which brings revisions to the media package with usability enhancements, such as addition of the Listener model and reduction of event interfaces, to the core package)
- Avaya awaits the release of the JTAPI 1.4 specification, which will carry these changes out to the callcontrol and callcenter packages. The ECTF expects to release JTAPI 1.4 during the second half of 2001.

## How to Use This Document

The following list maps specific activities to specific parts of this document.

- **Programming with JTAPI for any switch for which there is a CentreVu Telephony Services driver** — refer to Chapter 1: [JTAPI for all switches and the DEFINITY ECS](#), and, optionally, Chapter 2: [Telephony Services Extensions to JTAPI](#)
  - **Programming with JTAPI for applications used with the DEFINITY Switch and the G3 PBX Driver (G3PD)**, refer to the following parts of this document  
[Chapter 1, JTAPI for all switches and the DEFINITY ECS](#)  
[Chapter 3: DEFINITY-Specific Extensions](#)  
optionally,  
[Chapter 4, Telephony Services Extensions to JTAPI](#)
  - **Programming with JTAPI and TSAPI (Switch-Independent)**
    - to produce private data packages for applications used with non-DEFINITY switches and their associated drivers — see [Private Data Extensions](#), and [JTAPI for Private Data](#)
    - to use or interpret private data for applications used with non-DEFINITY switches and their associated drivers — see [Private Data Extensions](#), and [JTAPI for Private Data](#)
- 

## For More Information

Refer to the following documents for background information.

- Java Telephony API (JTAPI) Programmer's Reference (JTAPI v1.2) on the CentreVu Computer Telephony CD-ROM.

This document consists of Sun Microsystems's Java Telephony API (JTAPI) specification files that are available to you from the Sun Microsystems 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 DEFINITY Enterprise Communications Server Programmer's Guide* (DEFPROG.PDF on the CVCT CD ROM). This document describes DEFINITY switch administration and switch interactions
- *Telephony Services Application Programming Interface (TSAPI) Version 2* (TSAPI.PDF on the CVCT CD-ROM). This document describes 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.

This section describes 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
- for the DEFINITY switch and the G3PD driver

JTAPI interfaces and methods are grouped into the packages listed below.

<a href="#">Core Package</a>	<a href="#">Call Control Capabilites Package</a>	Phone Package (not supported)
<a href="#">Call Center Package</a>	<a href="#">Call Control Events Package</a>	Phone Events Package (not supported)
<a href="#">Call Center Capabilities Package</a>	<a href="#">Core Capabilities Package</a>	<a href="#">Private Data Package</a>
<a href="#">Call Center Events Package</a>	<a href="#">Core Events Package</a>	<a href="#">Private Data Capabilities Package</a>
<a href="#">Call Control Package</a>	<a href="#">Media Package</a>	<a href="#">Private Data Events Package</a>

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 "[Telephony Services Private Data Extensions to JTAPI](#)."

### Support for JTAPI Core Package

JTAPI Interfaces and Methods	Switches Supported
<b>Address</b>	<b>All</b>
getName	<b>All</b>
getProvider	<b>All</b>
getTerminals <sup>5</sup>	<b>All</b>
getConnections	<b>All</b>
addObserver	<b>All</b>
getObservers	<b>All</b>
removeObserver	<b>All</b>
addCallObserver	<b>All</b>
getCallObservers	<b>All</b>
removeCallObserver	<b>All</b>
getAddressCapabilities	<b>All</b>
<b>AddressObserver</b>	<b>All</b>
addressChangedEvent	<b>All</b>
<b>Call</b>	<b>All</b>
getConnections	<b>All</b>

getProvider	All
getState	All
connect	All
addObserver	All
getObservers	All
removeObserver	All
getCallCapabilities	All
<b>CallObserver</b>	All
callChangedEvent	All
<b>Connection</b>	All
getState	All
getCall	All
getAddress	All
getTerminalConnections	All
disconnect <a href="#">1</a>	All
getConnectionCapabilities	All
<b>JtapiPeer <a href="#">2</a></b>	All
getName	All
getServices <a href="#">3</a>	All
getProvider <a href="#">4</a>	All
<b>Provider</b>	All
getState	All
getName	All
getCalls	All
getAddress	All
getAddresses	All
getTerminals	All

getTerminal	All
shutdown	All
createCall	All
addObserver	All
getObservers	All
removeObserver	All
getProviderCapabilities	All
getCallCapabilities	All
getConnectionCapabilities	All
getAddressCapabilities	All
getTerminalConnectionCapabilities	All
getTerminalCapabilities	All
<b>ProviderObserver</b>	All
providerChangedEvent	All
<b>Terminal</b>	All
getName	All
getProvider	All
getAddresses <sup>5</sup>	All
getTerminalConnections	All
addObserver	All
getObservers	All
removeObserver	All
addCallObserver	All
getCallObservers	All
removeCallObserver	All
getTerminalCapabilities	All
<b>TerminalConnection</b>	All

getState	All
getTerminal	All
getConnection	All
answer	All
getTerminalConnectionCapabilities	All
<b>TerminalObserver</b>	All
terminalChangedEvent	All

#### 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.avaya.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>
5. CentreVu Computer Telephony Release 9.5 allows for an instance of the Telephony Server (Tserver) without the Telephony Services Security Database (SDB). The absence of the SDB results in the return of a NULL for address.getTerminal and terminal.getAddress because JTAPI depends on the contents of the SDB. Without the SDB, there is no listing of addresses and no information to pass.

#### 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 (including DEFINITY ), DEFINITY only, or none (if the implementation is not supported for any switches).

JTAPI Interfaces and Methods	Switches Supported
<b>ACDAddress</b>	All
getLoggedOnAgents <a href="#">1</a>	All
getNumberQueued <a href="#">2</a>	All
getOldestCallQueued	None
getRelativeQueueLoad	None
getQueueWaitTime	None
getACDManagerAddress	None
<b>ACDAddressObserver</b>	All

<b>ACDConnection</b>	DEFINITY only
getACDManagerConnection	DEFINITY only
<b>ACDManagerAddress</b>	DEFINITY only
getACDAddresses	None
<b>ACDManagerConnection</b>	DEFINITY only
getACDConnections	DEFINITY only
<b>AgentObject</b>	All
setState	All
getState	All
getAgentID <a href="#">3</a>	All
getACDAddress	All
getAgentAddress	All
getAgentTerminal	All
<b>AgentTerminal</b>	All
addAgent	All
removeAgent	All
getAgents	All
<b>AgentTerminalObserver</b> <a href="#">4</a>	All
<b>CallCenterAddress</b>	All
addCallObserver	All
<b>CallCenterCall</b>	All
connectPredictive <a href="#">5</a>	All
setApplicationData	None
getApplicationData	None
getTrunks	All
<b>CallCenterCallObserver</b>	All



<b>CallCenterProvider</b>	All
getRouteableAddresses	All
getACDAddresses	DEFINITY only
getACDManagerAddresses	DEFINITY only
<b>CallCenterTrunk</b>	All
getName	All
getState	All
getType	All
getCall	All
<b>RouteAddress</b>	All
registerRouteCallback <a href="#">6</a>	All
cancelRouteCallback	All
getRouteCallback	All
getActiveRouteSessions	All
<b>RouteCallback</b>	All
routeEvent	All
reRouteEvent	All
routeUsedEvent	All
routeEndEvent	All
routeCallbackEndedEvent	All
<b>RouteSession</b>	All
getRouteAddress	All
selectRoute <a href="#">7</a>	All
endRoute	All
getState	All
getCause	All

Implementation Notes

1. The `ACDAddress/getLoggedOnAgents` method is fully supported for the `DEFINITY` switch. For other switches, it returns the sum of (a) those agents that were logged in through the application and (b) those agents that were logged in after an `ACDAddressObserver` was added to the application.
2. The `ACDAddress/getNumberQueued` method is fully supported for the `DEFINITY` switch. For other switches, it returns the number of calls queued reported in the last queued event. This may not be accurate since some of the calls may have been subsequently dequeued.
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). The implementation of all JTAPI call center capabilities is supported for all switches (as indicated by "all.")

JTAPI Interfaces and Methods	Supported for All Switches
<b>ACDAddressCapabilities</b>	<b>All</b>
<code>canGetLoggedOnAgents</code>	<b>All</b>
<code>canGetNumberQueued</code>	<b>All</b>
<code>canGetOldestCallQueued</code>	<b>All</b>
<code>canGetRelativeQueueLoad</code>	<b>All</b>
<code>canGetQueueWaitTime</code>	<b>All</b>
<code>canGetACDManagerAddress</code>	<b>All</b>
<b>ACDConnectionCapabilities</b>	<b>All</b>
<code>canGetACDManagerConnection</code>	<b>All</b>
<b>ACDManagerAddressCapabilities</b>	<b>All</b>

canGetACDAddresses	All
<b>ACDManagerConnectionCapabilities</b>	All
canGetACDConnections	All
<b>AgentTerminalCapabilities</b>	All
canHandleAgents	All
<b>CallCenterAddressCapabilities</b>	All
canAddCallObserver	All
<b>CallCenterCallCapabilities</b>	All
canConnectPredictive	All
canHandleApplicationData	All
canGetTrunks	All
<b>CallCenterProviderCapabilities</b>	All
canGetRouteableAddresses	All
canGetACDAddresses	All
canGetACDManagerAddresses	All
canGetTrunks	All
<b>RouteAddressCapabilities</b>	All
canRouteCalls	All

### 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 (including DEFINITY ), DEFINITY only, or none (if the implementation is not supported for any switches).

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

JTAPI Interfaces	Switches Supporte
<b>ACDAddrBusyEv</b>	DEFINITY only
<b>ACDAddrLoggedOffEv</b>	All
<b>ACDAddrLoggedOnEv</b>	All
<b>ACDAddrNotReadyEv</b> <a href="#">1</a>	All

<b>ACDAddrReadyEv</b> <a href="#">1</a>	All
<b>ACDAddrUnknownEv</b>	All
<b>ACDAddrWorkNotReadyEv</b> <a href="#">1</a>	All
<b>ACDAddrWorkReadyEv</b> <a href="#">1</a>	All
<b>AgentTermBusyEv</b>	DEFINITY only
<b>AgentTermLoggedOffEv</b>	All
<b>AgentTermLoggedOnEv</b>	All
<b>AgentTermNotReadyEv</b> <a href="#">1</a>	All
<b>AgentTermReadyEv</b> <a href="#">1</a>	All
<b>AgentTermUnknownEv</b>	All
<b>AgentTermWorkNotReadyEv</b> <a href="#">1</a>	All
<b>AgentTermWorkReadyEv</b> <a href="#">1</a>	All
<b>CallCentCallAppDataEv</b>	None
<b>RouteCallbackEndedEvent</b>	All
<b>RouteEndEvent</b>	All
<b>RouteEvent</b>	All
<b>RouteSessionEvent</b>	All
<b>RouteUsedEvent</b>	All

#### 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 (including DEFINITY ), DEFINITY only, or none (if the implementation is not supported for any switches).

JTAPI Interfaces and Methods	Switches Supported
<b>CallControlAddress</b>	All
setForwarding <a href="#">1</a>	All

getForwarding	All
cancelForwarding	All
getDoNotDisturb <a href="#">2</a>	All
setDoNotDisturb <a href="#">2</a>	All
getMessageWaiting	All
setMessageWaiting	All
<b>CallControlAddressObserver</b>	All
<b>CallControlCall</b>	All
getCallingAddress	All
getCallingTerminal	All
getCalledAddress	All
getLastRedirectedAddress	All
addParty	DEFINITY only
drop	All
offHook	None
conference	All
transfer(Call otherCall)	All
transfer(String address)	None
setConferenceController	All
getConferenceController	All
setTransferController	All
getTransferController	All
setConferenceEnable	All
getConferenceEnable	All
setTransferEnable	All
getTransferEnable	All
consult (TerminalConnection termconn, String address)	All

consult (TerminalConnection termconn)	None
<b>CallControlCallObserver</b>	<b>All</b>
<b>CallControlConnection</b>	<b>All</b>
getCallControlState	<b>All</b>
accept	None
reject	None
redirect	<b>All</b>
addToAddress	None
park	None
<b>CallControlTerminal</b>	<b>All</b>
getDoNotDisturb <a href="#">2</a>	<b>All</b>
setDoNotDisturb <a href="#">2</a>	<b>All</b>
pickup (Connection pickConnection, Address terminalAddress)	<b>All</b>
pickup (TerminalConnection pickTermConn, Address terminalAddress)	<b>All</b>
pickup (Address pickAddress, Address terminalAddress)	<b>All</b>
pickupFromGroup(String pickupGroup, Address terminalAddress)	None
pickupFromGroup(Address terminalAddress)	<b>All</b>
<b>CallControlTerminalConnection</b>	<b>All</b>
getCallControlState	<b>All</b>
hold	<b>All</b>
unhold	<b>All</b>
join	None
leave	None
<b>CallControlTerminalObserver</b>	<b>All</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.

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). The implementation of all JTAPI call control capabilities is supported for all switches (as indicated by "all.")

JTAPI Interfaces and Methods	Switches Supported
<b>CallControlAddressCapabilities</b>	<b>All</b>
canSetForwarding	<b>All</b>
canGetForwarding	<b>All</b>
canCancelForwarding	<b>All</b>
canGetDoNotDisturb	<b>All</b>
canSetDoNotDisturb	<b>All</b>
canGetMessageWaiting	<b>All</b>
canSetMessageWaiting	<b>All</b>
<b>CallControlCallCapabilities</b>	<b>All</b>
canDrop	<b>All</b>
canOffHook	<b>All</b>
canSetConferenceController	<b>All</b>
canSetTransferController	<b>All</b>
canSetTransferEnable	<b>All</b>
canSetConferenceEnable	<b>All</b>
canTransfer	<b>All</b>
canConference	<b>All</b>
canAddParty	<b>All</b>

canConsult	All
<b>CallControlConnectionCapabilities</b>	All
canRedirect	All
canAddToAddress	All
canAccept	All
canReject	All
canPark	All
<b>CallControlTerminalCapabilities</b>	All
canGetDoNotDisturb	All
canSetDoNotDisturb	All
canPickup	All
canPickupFromGroup	All
<b>CallControlTerminalConnectionCapabilities</b>	All
canHold	All
canUnhold	All
canJoin	All
canLeave	All

#### Support for JTAPI Call Control Events Package

The following table lists each JTAPI interface from the JTAPI Call Control Events Package, (e.g., **CallCtIAddrDoNotDisturbEv**, **CallCtIAddrForwardEv**, **CallCtIAddrMessageWaitingEv**, and so forth), and whether the implementation is supported for all switches (including DEFINITY ), DEFINITY only, or none (if the implementation is not supported for any switches).

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

JTAPI Interfaces	Switches Supported
<b>CallCtIAddrDoNotDisturbEv</b> <a href="#">1</a>	All
<b>CallCtIAddrForwardEv</b>	All
<b>CallCtIAddrMessageWaitingEv</b>	All
<b>CallCtIConnAlertingEv</b>	All
<b>CallCtIConnDialingEv</b>	None



<b>CallCtlConnDisconnectedEv</b>	<b>All</b>
<b>CallCtlConnEstablishedEv</b>	<b>All</b>
<b>CallCtlConnFailedEv</b>	<b>All</b>
<b>CallCtlConnInitiatedEv</b>	<b>All</b>
<b>CallCtlConnNetworkAlertingEv</b>	<b>All</b>
<b>CallCtlConnNetworkReachedEv</b>	<b>All</b>
<b>CallCtlConnOfferedEv</b>	None
<b>CallCtlConnQueuedEv</b>	<b>All</b>
<b>CallCtlConnUnknownEv</b>	<b>All</b>
<b>CallCtlTermConnBridgedEv</b>	<b>All</b>
<b>CallCtlTermConnDroppedEv</b>	<b>All</b>
<b>CallCtlTermConnHeldEv</b>	<b>All</b>
<b>CallCtlTermConnInUseEv</b>	None
<b>CallCtlTermConnRingingEv</b>	<b>All</b>
<b>CallCtlTermConnTalkingEv</b>	<b>All</b>
<b>CallCtlTermConnUnknownEv</b>	<b>All</b>

#### Implementation Notes

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

#### Support for JTAPI Capabilities Package

The following table lists each JTAPI interface from the JTAPI Capabilities Package, (e.g., **AddressCapabilities**), followed by its associated method(s), (e.g., `isObservable`). The implementation of all JTAPI Interfaces and methods from the JTAPI capabilities package is supported for all switches (as indicated by "all.")

<b>JTAPI Interfaces and Methods</b>	<b>Switches Supported</b>
<b>AddressCapabilities</b>	<b>All</b>
<code>isObservable</code>	<b>All</b>
<b>CallCapabilities</b>	<b>All</b>
<code>canConnect</code>	<b>All</b>
<code>isObservable</code>	<b>All</b>

<b>ConnectionCapabilities</b>	All
canDisconnect	All
<b>ProviderCapabilities</b>	All
isObservable	All
<b>TerminalCapabilities</b>	All
isObservable	All
<b>TerminalConnectionCapabilities</b>	All
isObservable	All

### 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). The implementation of all JTAPI Interfaces and methods from the JTAPI Events package is supported for all switches (as indicated by "all.")

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

<b>JTAPI Interfaces</b>	<b>Supported for All Switches</b>
<b>AddObservationEndedEv</b>	All
<b>CallActiveEv</b>	All
<b>CallInvalidEv</b>	All
<b>CallObservationEndedEv</b>	All
<b>ConnAlertingEv</b>	All
<b>ConnConnectedEv</b>	All
<b>ConnCreatedEv</b>	All
<b>ConnDisconnectedEv</b>	All
<b>ConnFailedEv</b>	All
<b>ConnInProgressEv</b>	All
<b>ConnUnknownEv</b>	All

<b>ProvInServiceEv</b>	<b>All</b>
<b>ProvObservationEndedEv</b>	<b>All</b>
<b>ProvOutOfServiceEv</b>	<b>All</b>
<b>ProvShutdownEv</b>	<b>All</b>
<b>TermConnActiveEv</b>	<b>All</b>
<b>TermConnCreatedEv</b>	<b>All</b>
<b>TermConnDroppedEv</b>	<b>All</b>
<b>TermConnPassiveEv</b>	<b>All</b>
<b>TermConnRingingEv</b>	<b>All</b>
<b>TermConnUnknownEv</b>	<b>All</b>
<b>TermObservationEndedEv</b>	<b>All</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 (including DEFINITY ), DEFINITY only, or none (if the implementation is not supported for any switches).

<b>JTAPI Interfaces and Methods</b>	<b>Switches Supported</b>
<b>MediaCallObserver</b>	DEFINITY only
<b>MediaTerminalConnection</b>	DEFINITY only
getMediaAvailability	None
getMediaState	None
useDefaultSpeaker	None
useRecordURL	None
useDefaultMicrophone	None
usePlayURL	None
startPlaying	None

stopPlaying	None
startRecording	None
setDtmfDetection	None
generateDtmf	DEFINITY only

#### 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). The implementation of all JTAPI Interfaces and methods from the JTAPI Media Capabilities package is supported for all switches (as indicated by "all.")

JTAPI Interfaces and Methods	Switches Supported
<b>MediaTerminalConnectionCapabilities</b>	<b>All</b>
canUseDefaultSpeaker	<b>All</b>
canUseDefaultMicrophone	<b>All</b>
canUseRecordURL	<b>All</b>
canUsePlayURL	<b>All</b>
canStartPlaying	<b>All</b>
canStopPlaying	<b>All</b>
canStartRecording	<b>All</b>
canStopRecording	<b>All</b>
canDetectDtmf	<b>All</b>
canGenerateDtmf	<b>All</b>

#### Support for JTAPI Media Events Package

The following table lists each JTAPI interface from the JTAPI Media Events Package, (e.g., **MediaTermConnAvailable**, **MediaTermConnDtmfEv**, **MediaTermConnEv**, and so forth), and whether the implementation is supported for all switches (including DEFINITY ), DEFINITY only, or none (if the implementation is not supported for any switches).

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

JTAPI Interfaces	Supported for All Switches
<b>MediaTermConnAvailableEv</b>	None
<b>MediaTermConnDtmfEv</b> <sup>1</sup>	DEFINITY only

<b>MediaTermConnStateEv</b>	None
<b>MediaTermConnUnavailableEv</b>	None

#### 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

The JTAPI Phone Package interfaces and methods are not supported.

#### Support for JTAPI Phone Capabilities Package

The JTAPI Phone Package interfaces and methods are not supported.

#### Support for JTAPI Phone Events Package

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

If a JTAPI Phone Event is supported, all associated methods are also 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). The implementation of all JTAPI Interfaces and methods from the JTAPI Private Data package is supported for all switches (as indicated by "all.")

<b>JTAPI Interfaces and Methods</b>	<b>Switches Supported</b>
<b>PrivateData</b>	<b>All</b>
getPrivateData	<b>All</b>
setPrivateData 1	<b>All</b>
sendPrivateData 1	<b>All</b>

#### 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). The implementation of all JTAPI Interfaces and methods from the JTAPI Private Data Capabilities package is supported for all switches (as indicated by "all.")

<b>JTAPI Interfaces and Methods</b>	<b>Supported for All Switches</b>
-------------------------------------	-----------------------------------

<b>PrivateDataCapabilities</b>	<b>All</b>
canSetPrivateData	<b>All</b>
canGetPrivateData	<b>All</b>
canSendPrivateData	<b>All</b>

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). The implementation of all JTAPI Interfaces and methods from the JTAPI Private Data Events package is supported for all switches (as indicated by "all.")

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

<b>JTAPI Interfaces</b>	<b>Switches Supported</b>
<b>PrivateAddrEv</b>	<b>All</b>
<b>PrivateCallEv</b>	<b>All</b>
<b>PrivateProvEv</b>	<b>All</b>
<b>PrivateTermEv</b>	<b>All</b>

# Chapter 2

## Telephony Services Extensions to JTAPI for All Switches

### Using Telephony Services Extensions to JTAPI

The information on this page applies to 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 information 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.

If you are an application programmer who is using JTAPI to develop applications for the DEFINITY switch, also refer to "[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 "[Telephony Services DEFINITY-Specific Extensions to JTAPI](#)."

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 "[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.

For information about Computer-Supported Telecommunications Applications (CSTA) and API Control Services (ACS) error codes, refer to *Centre Vu Computer Telephony Telephony Services Application Programming Interface (TSAPI) Version 2* (TSAPI.PDF on the CVCT CD ROM).

#### 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:

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.avaya.jtapi.tsapi

## Interface Index

- [ITsapiException](#)

## Class Index

## Exception Index

- [TsapiInvalidArgumentException](#)
- [TsapiInvalidPartyException](#)
- [TsapiInvalidStateException](#)
- [TsapiMethodNotSupportedException](#)
- [TsapiPlatformException](#)
- [TsapiPrivilegeViolationException](#)
- [TsapiProviderUnavailableException](#)
- [TsapiResourceUnavailableException](#)



---

# Interface com.avaya.jtapi.tsapi.ITsapiException

public abstract 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

### ■ [ACS](#)

Error Type of ACS.

### ■ [CSTA](#)

Error Type of CSTA.

### ■ [EC\\_INVALID\\_CONF](#)

Error Code implying confirmation is invalid.

### ■ [EC\\_NORMAL](#)

Error Code of NORMAL.

### ■ [EC\\_PROVIDER\\_OUT\\_OF\\_SERVICE](#)

Error Code implying Provider is OUT\_OF\_SERVICE.

### ■ [INTERNAL](#)

Failure is internal to this Jtapi implementation.

### ■ [JTAPl](#)

Failed to meet some Jtapi condition.

### ■ [NORMAL](#)

Error Type of Normal.

## Method Index

### ■ [getErrorCode\(\)](#)

Returns the error code.

### ■ [getErrorType\(\)](#)

Returns the error type.

## Variables

### ■ ACS

```
public static final int ACS
```

Error Type of ACS.

## **CSTA**

```
public static final int CSTA
```

Error Type of CSTA.

## **EC\_INVALID\_CONF**

```
public static final int EC_INVALID_CONF
```

Error Code implying confirmation is invalid.

## **EC\_NORMAL**

```
public static final int EC_NORMAL
```

Error Code of NORMAL.

## **EC\_PROVIDER\_OUT\_OF\_SERVICE**

```
public static final int EC_PROVIDER_OUT_OF_SERVICE
```

Error Code implying Provider is OUT\_OF\_SERVICE.

## **INTERNAL**

```
public static final int INTERNAL
```

Failure is internal to this Jtapi implementation.

## **JTAPI**

```
public static final int JTAPI
```

Failed to meet some Jtapi condition.

## **NORMAL**

```
public static final int NORMAL
```

Error Type of Normal.

# Methods

## **getErrorCode**

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

Returns the error code.

## **getErrorType**

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

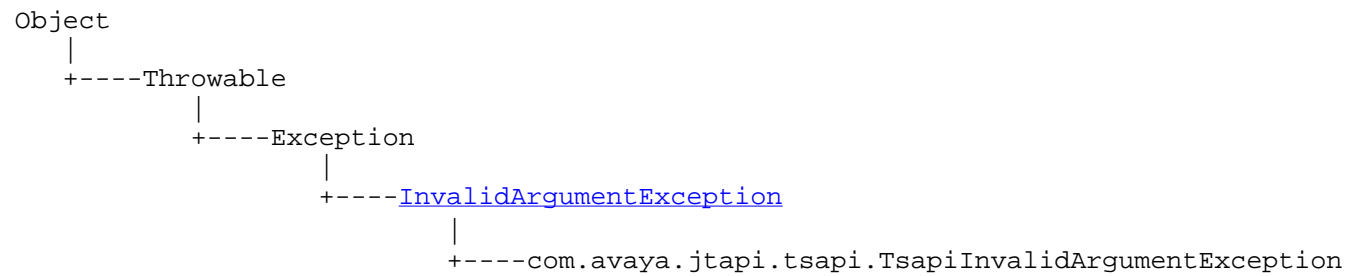
Returns the error type.

---

---

# Class

## com.avaya.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

### ▪ [getErrorCode\(\)](#)

Returns the error code.

### ▪ [getErrorType\(\)](#)

Returns the error type.

## Methods

### 🔴 **getErrorCode**

```
public int getErrorCode()
```

Returns the error code.

### 🔴 **getErrorType**

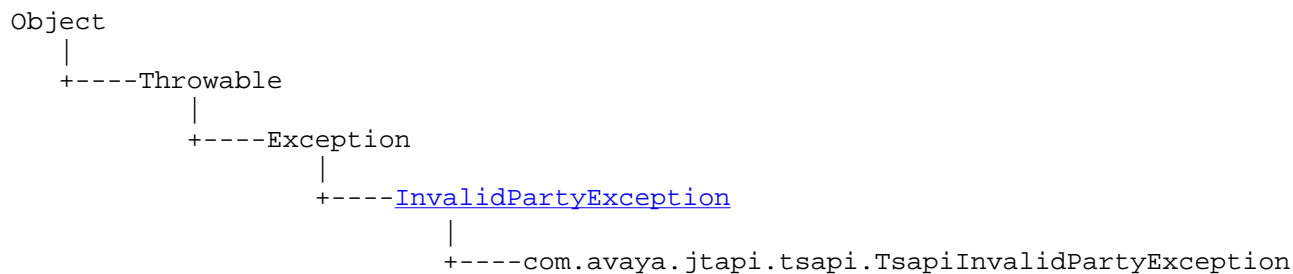
```
public int getErrorType()
```

Returns the error type.

---

---

# Class com.avaya.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

### ▪ [getErrorCode\(\)](#)

Returns the error code.

### ▪ [getErrorType\(\)](#)

Returns the error type.

## Methods

### 🔴 **getErrorCode**

```
public int getErrorCode()
```

Returns the error code.

### 🔴 **getErrorType**

```
public int getErrorType()
```

Returns the error type.

---

---

# Class com.avaya.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*

### ▪ [getErrorCode\(\)](#)

Returns the error code.

### ▪ [getErrorType\(\)](#)

Returns the error type.

## *Methods*

### ● **getErrorCode**

```
public int getErrorCode()
```

Returns the error code.

### ● **getErrorType**

```
public int getErrorType()
```

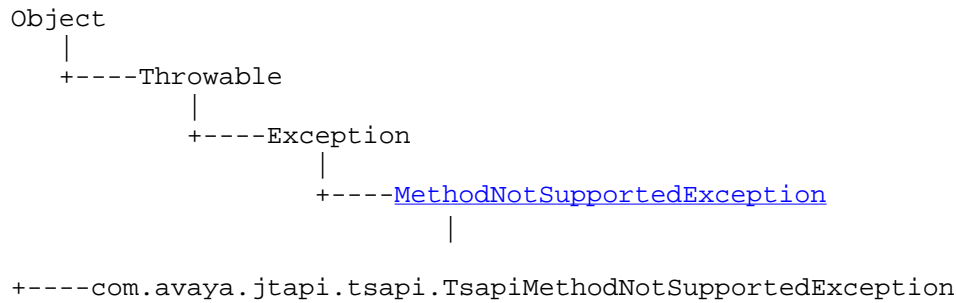
Returns the error type.

---

---

# Class

## com.avaya.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

### ▪ [getErrorCode\(\)](#)

Returns the error code.

### ▪ [getErrorType\(\)](#)

Returns the error type.

## Methods

### ● **getErrorCode**

```
public int getErrorCode()
```

Returns the error code.

### ● **getErrorType**

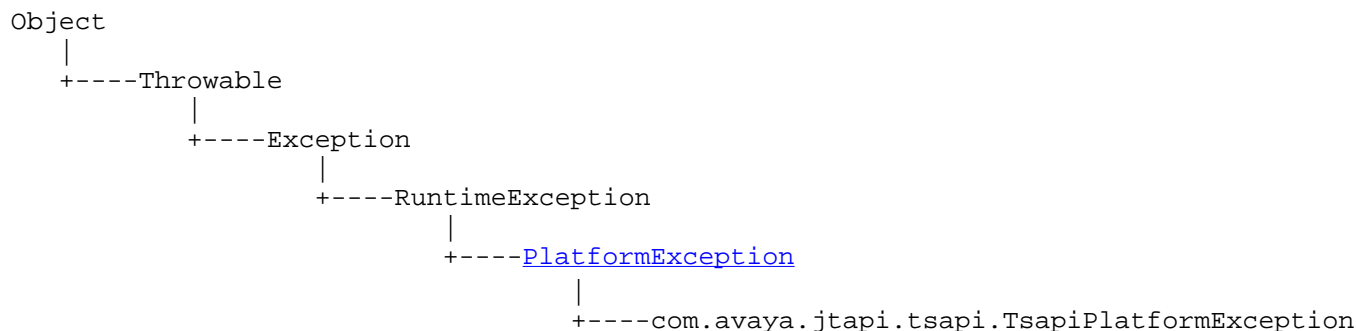
```
public int getErrorType()
```

Returns the error type.

---

---

# Class com.avaya.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

### ▪ [getErrorCode\(\)](#)

Returns the error code.

### ▪ [getErrorType\(\)](#)

Returns the error type.

## Methods

### ● **getErrorCode**

```
public int getErrorCode()
```

Returns the error code.

### ● **getErrorType**

```
public int getErrorType()
```

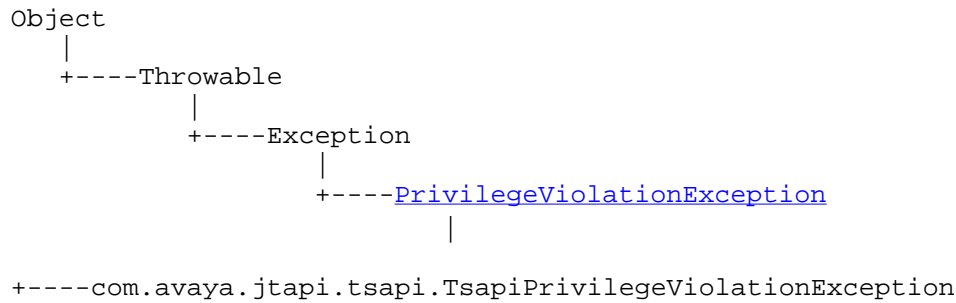
Returns the error type.

---

---

# Class

## com.avaya.jtapi.tsapi.TsapiPrivilegeViolationException



---

public final class **TsapiPrivilegeViolationException**

extends [PrivilegeViolationException](#)

implements [ITsapiException](#)

TsapiPrivilegeViolationException extends PrivilegeViolationException to add acs / csta error codes.

---

## Method Index

### ▪ [getErrorCode\(\)](#)

Returns the error code.

### ▪ [getErrorType\(\)](#)

Returns the error type.

## Methods

### ● **getErrorCode**

```
public int getErrorCode()
```

Returns the error code.

### ● **getErrorType**

```
public int getErrorType()
```

Returns the error type.

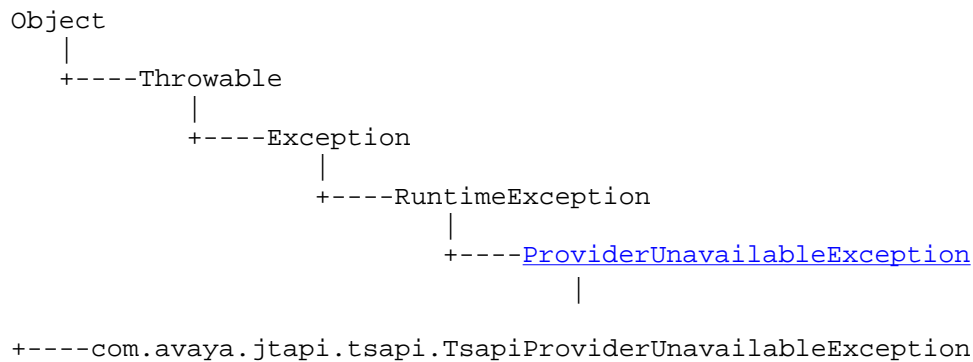
---



---

# Class

## com.avaya.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

### ▪ [getErrorCode\(\)](#)

Returns the error code.

### ▪ [getErrorType\(\)](#)

Returns the error type.

## Methods

### ● **getErrorCode**

```
public int getErrorCode()
```

Returns the error code.

### ● **getErrorType**

```
public int getErrorType()
```

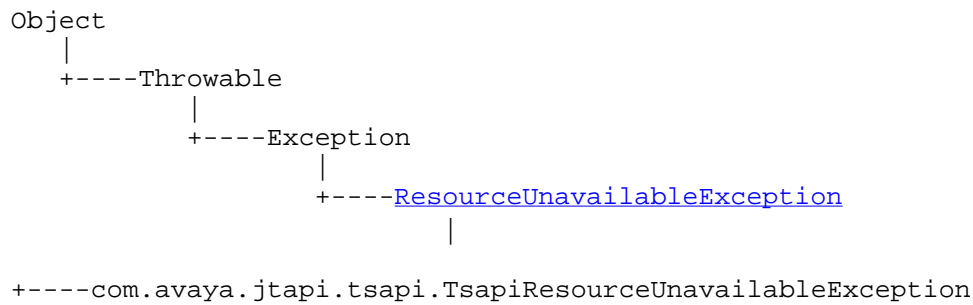
Returns the error type.

---

---

# Class

## com.avaya.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

### ▪ [getErrorCode\(\)](#)

Returns the error code.

### ▪ [getErrorType\(\)](#)

Returns the error type.

## Methods

### ● **getErrorCode**

```
public int getErrorCode()
```

Returns the error code.

### ● **getErrorType**

```
public int getErrorType()
```

Returns the error type.

---

# Chapter 3

## Telephony Services DEFINITY-Specific Extensions

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 information 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 , [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 [Using Telephony Services Extensions to JTAPI](#).

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 [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.

## DEFINITY Features Provided by DEFINITY-Specific Extensions to JTAPI

This table lists each DEFINITY feature that is available as an extension to JTAPI, its description, its associated class or interface, and the methods returned or used by methods in each appropriate class or interface.

Feature Name and Description	Class or Interface	Returned/Used by Methods in Class or Interface
<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.  G3V6 adds support for Pending Work Modes. A JTAPI Application may request to change an Agent's state to Agent.WORK_NOT_READY and Agent.NOT_READY, and to have the state change be held "pending" until all current calls active on the Agent's Agent Terminal are completed.		LucentAgent LucentAgentStateInfo LucentTerminal LucentV5Terminal LucentV5TerminalEx LucentV5AgentStateInfo LucentV6Agent LucentV6AgentStateInfo

<p><b>Call Classifier Information</b> Provides information on call classifier port usage</p>	<p>CallClassifierInfo</p>	<p>LucentProvider</p>
<p><b>Collect Digits</b> Allows a route request to wait for a specified number of digits to be collected</p>		<p>LucentRouteSession</p>
<p><b>Dial-Ahead Digits</b> Allows a route request to place digits in a dial-ahead buffer</p>		<p>LucentRouteSession</p>
<p><b>Direct Agent Calls</b> Allows calls to be made to and from specific logged-in ACD Agents</p>		<p>LucentCall LucentCallEx LucentRouteSession</p>
<p><b>Dropping Resources</b> Allows specific switch resources to be dropped from the call</p>		<p>LucentConnection LucentTerminalConnection</p>
<p><b>Flexible Billing</b> Allows changing the billing rate for incoming 900-type calls (supported by G3V5)</p>		<p>LucentV5Call</p>
<p><b>Flexible Generation of DTMF Tones</b> enables an application to specify tone duration and inter-tone delay duration.</p>		<p>LucentV5TerminalConnectionEx</p>
<p><b>Integrated Directory Name</b> Allows the G3 Integrated Directory Database name to be returned</p>		<p>LucentAddress LucentTerminal</p>
<p><b>Look-Ahead Interflow Information</b> May be used by a routing server application to determine the proper destination of a call</p>	<p>LookaheadInfo</p>	<p>LucentCallInfo LucentCallInfo</p>
<p>Lucent <b>Call Information</b> Provides Avaya ECS -specific call information on Call and CallControlCall events; information includes delivering ACD, distributing Address, originating Trunk, reason for last Call event, and other information.</p> <p>G3V5 adds support for: Universal Call ID, Originator Type, and Flex Billing Flag.</p>	<p>LucentCallInfo LucentV5CallInfo LucentCallInfo LucentV5CallInfo</p>	<p>Implemented by Lucent call objects, route session objects, and CallControlCall events.LucentCallInfo (extended by LucentCall; extended by LucentV5CallInfo; extended by CallControlCall events)</p>

<p><b>Message Waiting Application Information</b> Indicates which types of applications have enabled message waiting</p>		<p>LucentAddress LucentAddressMsgWaitingEvent</p>
<p><b>Network Progress Information</b> Contains supplementary call progress information from the ISDN Progress Indicator Information Element</p> <p>G3V5 adds support for: trunk.</p>	<p>NetworkProgressInfo</p> <p>V5NetworkProgressInfo</p>	<p>LucentConnNetworkReachedEvent</p>
<p><b>Original Call Information</b> Contains information about the original call in conjunction with the Call.consult() service.</p> <p>G3V5 adds support for: Universal Call ID, Originator Type, and Flex Billing Flag.</p>	<p>OriginalCallInfo</p> <p>V5OriginalCallInfo</p>	<p>LucentCallInfo</p> <p>LucentV5CallInfo</p>
<p><b>Priority Calls</b> Enables priority calling</p>		<p>LucentCall</p> <p>LucentRouteSession</p>
<p><b>Selective Listen</b> Allows control of listen paths between parties on a conference call (supported by G3V5)</p>		<p>LucentV5TerminalConnection</p>
<p><b>Single Step Conference</b> Adds another party to a call (added party does not alert; used mainly for service observing) (supported by G3V5)</p>		<p>LucentV5Call</p>
<p><b>Supervisor Assist Calls</b> Allows logged-in ACD Agents to place calls to a supervisor's extension</p>		<p>LucentCall</p>
<p><b>Switch Date and Time Information</b> Returns the current date and time from the switch</p>		<p>LucentProvider</p>
<p><b>Trunk Group Information</b> Provides information on trunk group usage</p> <p>Trunk associates group and member information with a connection. If a connection is associated with a trunk party, then the application can get trunk group number and trunk group member information.</p>	<p>TrunkGroupInfo</p>	<p>LucentProvider</p> <p>LucentV6 Connection</p> <p>LucentTrunk</p> <p>ITsapiTrunk</p>
<p><b>Universal Call ID</b> A call identifier that is globally unique across switches and the public network (supported by G3V5)</p>		<p>LucentV5CallInfo (extended by LucentV5Call)</p>
<p><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</p>	<p>UserEnteredCode</p>	<p>LucentCallInfo</p> <p>OriginalCallInfo</p>

**User-to-User Information** An ISDN feature that allows end-to-end transmission of application data during call setup/teardown. UUI can be specified, and will be made available, accommodating string values up to 96 characters long.

UserToUserInfo

LucentCall

LucentCallInfo

LucentConnection

LucentRouteSession

LucentTerminalConnection

LucentCallInfo

---

# package com.avaya.jtapi.tsapi

## Interface Index

- [ITsapiAddress](#)
- [ITsapiAddressMsgWaitingEvent](#)
- [ITsapiAgent](#)
- [ITsapiCall](#)
- [ITsapiCallInfo](#)
- [ITsapiConnNetworkReachedEvent](#)
- [ITsapiConnection](#)
- [ITsapiRouteSession](#)
- [ITsapiTerminal](#)
- [ITsapiTerminalConnection](#)
- [ITsapiTrunk](#)
- [LucentAddress](#)
- [LucentAddressMsgWaitingEvent](#)
- [LucentAgent](#)
- [LucentCall](#)
- [LucentCallEx](#)
- [LucentCallInfo](#)
- [LucentConnNetworkReachedEvent](#)
- [LucentConnection](#)
- [LucentProvider](#)
- [LucentRouteSession](#)
- [LucentTerminal](#)
- [LucentTerminalConnection](#)
- [LucentTrunk](#)
- [LucentV5Call](#)
- [LucentV5CallInfo](#)
- [LucentV5Connection](#)
- [LucentV5Provider](#)
- [LucentV5Terminal](#)
- [LucentV5TerminalConnection](#)
- [LucentV5TerminalEx](#)
- [LucentV6Agent](#)
- [LucentV6Connection](#)

# Class Index

- [CallClassifierInfo](#)
- [LookaheadInfo](#)
- [LucentAgentStateInfo](#)
- [LucentBillType](#)
- [LucentChargeAdviceEvent](#)
- [LucentChargeError](#)
- [LucentChargeType](#)
- [LucentV5AgentStateInfo](#)
- [LucentV6AgentStateInfo](#)
- [NetworkProgressInfo](#)
- [OriginalCallInfo](#)
- [TrunkGroupInfo](#)
- [UserEnteredCode](#)
- [UserToUserInfo](#)
- [V5NetworkProgressInfo](#)
- [V5OriginalCallInfo](#)

# Exception Index



---

# Interface com.avaya.jtapi.tsapi.ITsapiAddress

public abstract 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.avaya.jtapi.tsapi.ITsapiAddressMsgWaitingEvent

public abstract 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.avaya.jtapi.tsapi.ITsapiAgent

public abstract 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.avaya.jtapi.tsapi.ITsapiCall

public abstract 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.avaya.jtapi.tsapi.ITsapiCallInfo

public abstract 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.avaya.jtapi.tsapi.ITsapiConnNetworkReachedEvent

public abstract 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.avaya.jtapi.tsapi.ITsapiConnection

public abstract 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.avaya.jtapi.tsapi.ITsapiRouteSession

public abstract interface **ITsapiRouteSession**

extends [RouteSession](#), [ITsapiCallInfo](#)

ITsapiRouteSession extends Jtapi RouteSession to return the Call object associated with the RouteSession.

---

## Method Index

### ▪ [getCall\(\)](#)

Returns the Call object associated with this RouteSession.

## Methods

### ▪ [getCall](#)

```
public abstract javax.telephony.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.avaya.jtapi.tsapi.ITsapiTerminal

public abstract 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.avaya.jtapi.tsapi.ITsapiTerminalConnection

public abstract 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.avaya.jtapi.tsapi.ITsapiTrunk

public abstract interface **ITsapiTrunk**

extends [CallCenterTrunk](#)

ITsapiTrunk extends Jtapi CallCenterTrunk.

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

---

---

---

# Interface com.avaya.jtapi.tsapi.LucentAddress

public abstract interface **LucentAddress**

extends [ITsapiAddress](#)

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

---

## Variable Index

### ▪ [MWI\\_CTI](#)

The message waiting indicator has been enabled via CTI.

### ▪ [MWI\\_LWC](#)

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

### ▪ [MWI\\_MCS](#)

The message waiting indicator has been enabled via Message Center.

### ▪ [MWI\\_PROPMGT](#)

The message waiting indicator has been enabled via Property Management.

### ▪ [MWI\\_VOICE](#)

The message waiting indicator has been enabled via Voice Messaging.

## Method Index

### ▪ [getDirectoryName\(\)](#)

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

### ▪ [getMessageWaitingBits\(\)](#)

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

## Variables

### • MWI\_CTI

```
public static final int MWI_CTI
```

The message waiting indicator has been enabled via CTI.

### • MWI\_LWC

```
public static final int MWI_LWC
```

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

### • MWI\_MCS

```
public static final int MWI_MCS
```

The message waiting indicator has been enabled via Message Center.

## MWI\_PROPMGT

```
public static final int MWI_PROPMGT
```

The message waiting indicator has been enabled via Property Management.

## MWI\_VOICE

```
public static final int MWI_VOICE
```

The message waiting indicator has been enabled via Voice Messaging.

# Methods

## **getDirectoryName**

```
public abstract java.lang.String getDirectoryName()
```

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

## **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.

---

---

# Interface

## com.avaya.jtapi.tsapi.LucentAddressMsgWaitingEvent

public abstract interface **LucentAddressMsgWaitingEvent**

extends [ITsapiAddressMsgWaitingEvent](#)

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

---

## *Method Index*

### ▪ [getMessageWaitingBits\(\)](#)

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

## *Methods*

### • [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.avaya.jtapi.tsapi.LucentAgent

public abstract interface **LucentAgent**

extends [ITsapiAgent](#)

The LucentAgent interface extends the ITsapiAgent interface.

---

## Variable Index

### ■ [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 .

### ■ [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 .

### ■ [MODE\\_NONE](#)

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

## Method Index

### ■ [getStateInfo\(\)](#)

This returns this Agent's state and workMode.

### ■ [setState\(int, int\)](#)

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

## Variables

### ● [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 .

### ● [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 .

### ● [MODE\\_NONE](#)

```
public static final int MODE_NONE
```

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

# Methods

## getStateInfo

```
public abstract com.avaya.jtapi.tsapi.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.

## 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.

---





boolean priorityCall,  
[UserToUserInfo](#) userInfo) throws

[TsapiResourceUnavailableException](#), [TsapiPrivilegeViolationException](#),  
[TsapiInvalidPartyException](#), [TsapiInvalidArgumentException](#),  
[TsapiInvalidStateException](#), [TsapiMethodNotSupportedException](#)

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

**Parameters:**

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

**See Also:**

[UserToUserInfo](#)

 **connectDirectAgent**

```
public abstract javax.telephony.Connection[] connectDirectAgent(LucentTerminal  
origterm,  
LucentAddress  
origaddr,  
LucentAgent  
calledAgent,  
boolean  
priorityCall,  
UserToUserInfo  
userInfo) throws TsapiResourceUnavailableException,  
TsapiPrivilegeViolationException, TsapiInvalidPartyException,  
TsapiInvalidArgumentException, TsapiInvalidStateException,  
TsapiMethodNotSupportedException
```

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](#)

 **connectPredictive**

```
public abstract javax.telephony.Connection[] connectPredictive(LucentTerminal  
originatorTerminal,  
LucentAddress  
origAddress,  
String dialedDigits,  
int connectionState,  
int maxRings,  
int  
answeringTreatment,  
int  
answeringEndpointType,  
boolean priorityCall,
```

userInfo) throws [TsapiResourceUnavailableException](#),  
[TsapiPrivilegeViolationException](#), [TsapiInvalidPartyException](#),  
[TsapiInvalidArgumentException](#), [TsapiInvalidStateException](#),  
[TsapiMethodNotSupportedException](#)

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](#)

 **connectSupervisorAssist**

```
public abstract javax.telephony.Connection[] connectSupervisorAssist(LucentAgent
callingAgent,
String
dialedDigits,
UserToUserInfo
userInfo) throws TsapiResourceUnavailableException,
TsapiPrivilegeViolationException, TsapiInvalidPartyException,
TsapiInvalidArgumentException, TsapiInvalidStateException,
TsapiMethodNotSupportedException
```

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](#)

 **consult**

```
public abstract javax.telephony.Connection[] consult(LucentTerminalConnection
termconn,
String address,
boolean priorityCall,
UserToUserInfo userInfo) throws
TsapiInvalidStateException, TsapiInvalidArgumentException,
TsapiMethodNotSupportedException, TsapiResourceUnavailableException,
```

## [TsapiPrivilegeViolationException](#)

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](#)

## **consultDirectAgent**

```
public abstract javax.telephony.Connection[]
consultDirectAgent(LucentTerminalConnection termconn,
                  LucentAgent
calledAgent,
                  boolean
priorityCall,
                  UserToUserInfo
userInfo) throws TsapiInvalidStateException, TsapiInvalidArgumentException,
TsapiMethodNotSupportedException, TsapiResourceUnavailableException,
TsapiPrivilegeViolationException
```

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](#)

## **consultSupervisorAssist**

```
public abstract javax.telephony.Connection[]
consultSupervisorAssist(LucentTerminalConnection termconn,
                       ACDAddress
split,
                       String address,
                       UserToUserInfo
userInfo) throws TsapiInvalidStateException, TsapiInvalidArgumentException,
TsapiMethodNotSupportedException, TsapiResourceUnavailableException,
TsapiPrivilegeViolationException
```

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.avaya.jtapi.tsapi.LucentCallEx

public abstract interface **LucentCallEx**

extends [LucentCall](#)

---

## Method Index

■ [connectDirectAgent](#)(LucentTerminal, LucentAddress, LucentAgent, boolean, UserToUserInfo, ACDAddress)

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

■ [consultDirectAgent](#)(LucentTerminalConnection, LucentAgent, boolean, UserToUserInfo, ACDAddress)

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

## Methods

### ● **connectDirectAgent**

```
public abstract javax.telephony.Connection[] connectDirectAgent(LucentTerminal
origterm,
LucentAddress
origaddr,
LucentAgent
calledAgent,
boolean
priorityCall,
UserToUserInfo
userInfo,
ACDAddress
acdaddress) throws TsapiResourceUnavailableException,
TsapiPrivilegeViolationException, TsapiInvalidPartyException,
TsapiInvalidArgumentException, TsapiInvalidStateException,
TsapiMethodNotSupportedException
```

Places a direct call to a specific logged-in ACD agent. This is a extension of the original connectDirectAgent which did not enable the user to specify ACDAddress.

#### 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.
- acdaddress - The ACDaddress used in the measuring of the direct agent call.

#### See Also:

[UserToUserInfo](#)

### ● **consultDirectAgent**

```
public abstract javax.telephony.Connection[]
consultDirectAgent(LucentTerminalConnection termconn,
                  LucentAgent
calledAgent,
                  boolean
priorityCall,
                  UserToUserInfo
userInfo,
                  ACDAddress
acdaddress) throws TsapiInvalidStateException, TsapiInvalidArgumentException,
TsapiMethodNotSupportedException, TsapiResourceUnavailableException,
TsapiPrivilegeViolationException
```

Places a consultation call with a specific logged-in ACD agent. This is an extension of the original `consultDirectAgent` which did not enable the user to specify `ACDAddress`.

**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.
- `acdaddress` - The `ACDAddress` used in the measuring of the direct agent call.

**See Also:**

[UserToUserInfo](#)

---

---

# Interface com.avaya.jtapi.tsapi.LucentCallInfo

public abstract 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

- [AR\\_ANSWER\\_MACHINE\\_DETECTED](#)
- [AR\\_ANSWER\\_NORMAL](#)
- [AR\\_ANSWER\\_TIMED](#)
- [AR\\_ANSWER\\_VOICE\\_ENERGY](#)
- [AR\\_IN\\_QUEUE](#)
- [AR\\_NONE](#)
- [AR\\_SIT\\_INEFFECTIVE\\_OTHER](#)
- [AR\\_SIT\\_INTERCEPT](#)
- [AR\\_SIT\\_NO\\_CIRCUIT](#)
- [AR\\_SIT\\_REORDER](#)
- [AR\\_SIT\\_UNKNOWN](#)
- [AR\\_SIT\\_VACANT\\_CODE](#)

## Method Index

### ▪ [getDeliveringACDAddress\(\)](#)

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

### ▪ [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.

### ▪ [getLookaheadInfo\(\)](#)

Returns lookahead interflow information associated with the call event.

### ▪ [getOriginalCallInfo\(\)](#)

Returns original call information associated with the call event.

### ▪ [getReason\(\)](#)

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

### ▪ [getTrunk\(\)](#)

Returns the trunk from which the call originated.

## ▪ [getUserEnteredCode\(\)](#)

Returns call prompting digits associated with the call event.

## ▪ [getUserToUserInfo\(\)](#)

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

# Variables

## • **AR\_ANSWER\_MACHINE\_DETECTED**

```
public static final short AR_ANSWER_MACHINE_DETECTED
```

## • **AR\_ANSWER\_NORMAL**

```
public static final short AR_ANSWER_NORMAL
```

## • **AR\_ANSWER\_TIMED**

```
public static final short AR_ANSWER_TIMED
```

## • **AR\_ANSWER\_VOICE\_ENERGY**

```
public static final short AR_ANSWER_VOICE_ENERGY
```

## • **AR\_IN\_QUEUE**

```
public static final short AR_IN_QUEUE
```

## • **AR\_NONE**

```
public static final short AR_NONE
```

## • **AR\_SIT\_INEFFECTIVE\_OTHER**

```
public static final short AR_SIT_INEFFECTIVE_OTHER
```

## • **AR\_SIT\_INTERCEPT**

```
public static final short AR_SIT_INTERCEPT
```

## • **AR\_SIT\_NO\_CIRCUIT**

```
public static final short AR_SIT_NO_CIRCUIT
```

## • **AR\_SIT\_REORDER**

```
public static final short AR_SIT_REORDER
```

## • **AR\_SIT\_UNKNOWN**

```
public static final short AR_SIT_UNKNOWN
```

## • **AR\_SIT\_VACANT\_CODE**

```
public static final short AR_SIT_VACANT_CODE
```



# Methods

## **getDeliveringACDAddress**

```
public abstract javax.telephony.callcenter.ACDAddress getDeliveringACDAddress()
```

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

## **getDistributingAddress**

```
public abstract javax.telephony.callcenter.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.

## **getLookaheadInfo**

```
public abstract com.avaya.jtapi.tsapi.LookaheadInfo getLookaheadInfo()
```

Returns lookahead interflow information associated with the call event.

## **getOriginalCallInfo**

```
public abstract com.avaya.jtapi.tsapi.OriginalCallInfo getOriginalCallInfo()
```

Returns original call information associated with the call event.

## **getReason**

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

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

## **getTrunk**

```
public abstract javax.telephony.callcenter.CallCenterTrunk getTrunk()
```

Returns the trunk from which the call originated.

## **getUserEnteredCode**

```
public abstract com.avaya.jtapi.tsapi.UserEnteredCode getUserEnteredCode()
```

Returns call prompting digits associated with the call event.

## **getUserToUserInfo**

```
public abstract com.avaya.jtapi.tsapi.UserToUserInfo getUserToUserInfo()
```

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

---

---

# Interface

## com.avaya.jtapi.tsapi.LucentConnNetworkReachedEvent

public abstract interface **LucentConnNetworkReachedEvent**

extends [ITsapiConnNetworkReachedEvent](#)

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

---

## *Method Index*

### ▪ [getNetworkProgressInfo\(\)](#)

Get the supplementary call progress information

## *Methods*

### ● **getNetworkProgressInfo**

public abstract com.avaya.jtapi.tsapi.NetworkProgressInfo getNetworkProgressInfo()

Get the supplementary call progress information

---

---

# Interface com.avaya.jtapi.tsapi.LucentConnection

public abstract 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

### ▪ [DR\\_CALL\\_CLASSIFIER](#)

Drop a call classifier from the call.

### ▪ [DR\\_NONE](#)

### ▪ [DR\\_TONE\\_GENERATOR](#)

Drop a tone generator from the call.

## Method Index

### ▪ [disconnect](#)(short, UserToUserInfo)

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

## Variables

### • **DR\_CALL\_CLASSIFIER**

```
public static final short DR_CALL_CLASSIFIER
```

Drop a call classifier from the call.

### • **DR\_NONE**

```
public static final short DR_NONE
```

### • **DR\_TONE\_GENERATOR**

```
public static final short DR_TONE_GENERATOR
```

Drop a tone generator from the call.

## Methods

### • **disconnect**

```
public abstract void disconnect(short dropResource,  
                               UserToUserInfo userInfo) throws  
TsapiPrivilegeViolationException, TsapiResourceUnavailableException,  
TsapiMethodNotSupportedException, TsapiInvalidStateException
```

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.avaya.jtapi.tsapi.LucentProvider

public abstract interface **LucentProvider**

extends [ITsapiProvider](#)

LucentProvider adds methods to obtain Lucent-specific switch information.

---

## Method Index

### ▪ [getCallClassifierInfo\(\)](#)

Returns information on call classifier port usage.

### ▪ [getSwitchDateAndTime\(\)](#)

Returns current date and time from the switch.

### ▪ [getTrunkGroupInfo\(String\)](#)

Returns trunk usage information on the specified trunk group.

## Methods

### ● **getCallClassifierInfo**

```
public abstract com.avaya.jtapi.tsapi.CallClassifierInfo getCallClassifierInfo()  
throws TsapiMethodNotSupportedException
```

Returns information on call classifier port usage.

### ● **getSwitchDateAndTime**

```
public abstract java.util.Date getSwitchDateAndTime() throws  
TsapiMethodNotSupportedException
```

Returns current date and time from the switch.

### ● **getTrunkGroupInfo**

```
public abstract com.avaya.jtapi.tsapi.TrunkGroupInfo getTrunkGroupInfo(String  
trunkAccessCode) throws TsapiMethodNotSupportedException
```

Returns trunk usage information on the specified trunk group.

---

---

# Interface com.avaya.jtapi.tsapi.LucentRouteSession

public abstract 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

▪ [selectRoute](#)(String, boolean, UserToUserInfo)

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

▪ [selectRouteAndCollect](#)(String, int, int, boolean, UserToUserInfo)

Routes a call and requests DTMF digit collection.

▪ [selectRouteDirectAgent](#)(LucentAgent, boolean, UserToUserInfo)

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

▪ [selectRouteWithDigits](#)(String, String, boolean, UserToUserInfo)

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

## Methods

### 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](#)

### selectRouteAndCollect

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

## [TsapiMethodNotSupportedException](#)

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](#)

## **selectRouteDirectAgent**

```
public abstract void selectRouteDirectAgent(LucentAgent calledAgent,  
                                             boolean priorityCall,  
                                             UserToUserInfo userInfo) throws  
TsapiMethodNotSupportedException, TsapiInvalidArgumentException
```

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](#)

## **selectRouteWithDigits**

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

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.avaya.jtapi.tsapi.LucentTerminal

public abstract interface **LucentTerminal**

extends [ITsapiTerminal](#)

The LucentTerminal interface extends the ITsapiTerminal interface.

---

## Method Index

▪ [addAgent](#)(LucentAddress, ACDAddress, int, int, String, String)

This method overrides Terminal.addAgent() to add the Lucent-specific parameter workMode.

▪ [getDirectoryName](#)()

Return Directory name of this Terminal.

## Methods

### addAgent

```
public abstract javax.telephony.callcenter.Agent addAgent(LucentAddress
agentAddress,
ACDAddress acdAddress,
int initialState,
int workMode,
String agentID,
String password) throws
TsapiInvalidArgumentException, TsapiInvalidStateException
```

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 ACDAAddress 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.

 **getDirectoryName**

```
public abstract java.lang.String getDirectoryName()
```

Return Directory name of this Terminal.

---

---

# Interface com.avaya.jtapi.tsapi.LucentTerminalConnection

public abstract 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

### ▪ [DR\\_CALL\\_CLASSIFIER](#)

Drop a call classifier from the call..

### ▪ [DR\\_NONE](#)

### ▪ [DR\\_TONE\\_GENERATOR](#)

Drop a tone generator from the call..

## Method Index

### ▪ [leave](#)(short, UserToUserInfo)

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

## Variables

### • **DR\_CALL\_CLASSIFIER**

```
public static final short DR_CALL_CLASSIFIER
```

Drop a call classifier from the call..

### • **DR\_NONE**

```
public static final short DR_NONE
```

### • **DR\_TONE\_GENERATOR**

```
public static final short DR_TONE_GENERATOR
```

Drop a tone generator from the call..

## Methods

### • **leave**

```
public abstract void leave(short dropResource,  
                           UserToUserInfo userInfo) throws  
TsapiInvalidStateException, TsapiMethodNotSupportedException,  
TsapiPrivilegeViolationException, TsapiResourceUnavailableException
```

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.avaya.jtapi.tsapi.LucentTrunk

public abstract interface **LucentTrunk**

extends [ITsapiTrunk](#)

This interface extends the ITsapiTrunk interface with features specific to DEFINITY G3 PBX Driver Version 6 private data. When a Provider is bound to a DEFINITY switch which supports V6 private data, this interface may be used to access additional capabilities.

This interface provides methods to access additional trunk information (member and group name) made available with DEFINITY R8 and the G3 PBX Driver private data version 6. It also provides access to the Connection associated with this trunk.

---

## Method Index

### ▪ [getConnection\(\)](#)

this method returns the Connection associated with this trunk, or null if this information is not available.

### ▪ [getGroupName\(\)](#)

this method returns the name of the trunk group for this trunk, or null if no trunk name or trunk group name is known.

### ▪ [getMemberName\(\)](#)

this method returns the name of the trunk member for this trunk, or null if no trunk name or trunk member name is known.

## Methods

### ● **getConnection**

```
public abstract javax.telephony.Connection getConnection()
```

this method returns the Connection associated with this trunk, or null if this information is not available.

#### **Returns:**

A Connection associated with this trunk, or null if this information is not available

### ● **getGroupName**

```
public abstract java.lang.String getGroupName()
```

this method returns the name of the trunk group for this trunk, or null if no trunk name or trunk group name is known.

#### **Returns:**

A String that is the name of the trunk group for this trunk, or null if no trunk name or trunk group name is known.

### ● **getMemberName**

```
public abstract java.lang.String getMemberName()
```

this method returns the name of the trunk member for this trunk, or null if no trunk name or trunk member name is known.

#### **Returns:**

A String that is the name of the trunk member for this trunk, or null if no trunk name or trunk member name is known

---

---

# Interface com.avaya.jtapi.tsapi.LucentV5Call

public abstract interface **LucentV5Call**

extends [ITsapiCall](#), [LucentCallEx](#), [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

▪ [addParty](#)(String, boolean)

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

▪ [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

• **addParty**

```
public abstract javax.telephony.Connection addParty(String newParty,
                                                    boolean isActive) throws
TsapiInvalidStateException, TsapiInvalidPartyException,
TsapiMethodNotSupportedException, TsapiPrivilegeViolationException,
TsapiResourceUnavailableException
```

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.

• **setBillRate**

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

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.avaya.jtapi.tsapi.LucentV5CallInfo

public abstract 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

### ■ [canSetBillRate\(\)](#)

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

### ■ [getCallOriginatorType\(\)](#)

Get the originator type for this call, such as coin call, 800 service call, or cellular call.

### ■ [getUCID\(\)](#)

Get the Universal Call ID for this call.

### ■ [hasCallOriginatorType\(\)](#)

Query whether CallOriginatorType is available for this call.

## Methods

### ● **canSetBillRate**

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

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

### ● **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

### **getUCID**

```
public abstract java.lang.String getUCID()
```

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

### **hasCallOriginatorType**

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

Query whether CallOriginatorType is available for this call.

---



---

# Interface com.avaya.jtapi.tsapi.LucentV5Connection

public abstract 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

### ■ [listenHold](#)(LucentTerminalConnection)

Places a non-bridged Connection's listening path on listen-hold with respect to the specified TerminalConnection.

### ■ [listenHold](#)(LucentConnection)

Places a non-bridged Connection's listening path on listen-hold with respect to the specified non-bridged Connection.

### ■ [listenUnhold](#)(LucentTerminalConnection)

Takes a non-bridged Connection's listening path off listen-hold with respect to the specified TerminalConnection.

### ■ [listenUnhold](#)(LucentConnection)

Takes a non-bridged Connection's listening path off listen-hold with respect to the specified non-bridged Connection.

## Methods

### ● **listenHold**

public abstract void listenHold([LucentTerminalConnection](#) partyToHold) throws

[TsapiInvalidStateException](#), [TsapiMethodNotSupportedException](#),  
[TsapiPrivilegeViolationException](#), [TsapiResourceUnavailableException](#),  
[TsapiInvalidArgumentException](#)

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.

### ● **listenHold**

public abstract void listenHold([LucentConnection](#) partyToHold) throws

[TsapiInvalidStateException](#), [TsapiMethodNotSupportedException](#),  
[TsapiPrivilegeViolationException](#), [TsapiResourceUnavailableException](#),

## [TsapiInvalidArgumentException](#)

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.

### **listenUnhold**

```
public abstract void listenUnhold(LucentTerminalConnection partyToUnhold) throws  
TsapiInvalidStateException, TsapiMethodNotSupportedException,  
TsapiPrivilegeViolationException, TsapiResourceUnavailableException,  
TsapiInvalidArgumentException
```

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.

### **listenUnhold**

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

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.avaya.jtapi.tsapi.LucentV5Provider

public abstract interface **LucentV5Provider**

extends [LucentProvider](#)

LucentV5Provider adds the Advice Of Charge feature.

---

## *Method Index*

▪ [setAdviceOfCharge](#)(boolean)

Activate or deactivate the Advice Of Charge feature.

## *Methods*

▪ [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.avaya.jtapi.tsapi.LucentV5Terminal

public abstract interface **LucentV5Terminal**

extends [LucentTerminal](#)

The LucentV5Terminal interface extends the LucentTerminal interface.

---

## Method Index

▪ [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

▪ **addAgent**

```
public abstract javax.telephony.callcenter.Agent addAgent(LucentAddress
agentAddress,
ACDAddress acdAddress,
int initialState,
int workMode,
int reasonCode,
String agentID,
String password) throws
```

[TsapiInvalidArgumentException](#), [TsapiInvalidStateException](#)

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 ACDAAddress 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.avaya.jtapi.tsapi.LucentV5TerminalConnection

public abstract 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

### ■ [listenHold](#)(`LucentTerminalConnection`)

Places a `TerminalConnection`'s listening path on listen-hold with respect to the specified `TerminalConnection`.

### ■ [listenHold](#)(`LucentConnection`)

Places a `TerminalConnection`'s listening path on listen-hold with respect to the specified non-bridged `Connection`.

### ■ [listenUnhold](#)(`LucentTerminalConnection`)

Takes a `TerminalConnection`'s listening path off listen-hold with respect to the specified `TerminalConnection`.

### ■ [listenUnhold](#)(`LucentConnection`)

Takes a `TerminalConnection`'s listening path off listen-hold with respect to the specified non-bridged `Connection`.

## Methods

### ● **listenHold**

```
public abstract void listenHold(LucentTerminalConnection partyToHold) throws
TsapiInvalidStateException, TsapiMethodNotSupportedException,
TsapiPrivilegeViolationException, TsapiResourceUnavailableException,
TsapiInvalidArgumentException
```

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.

### ● **listenHold**

```
public abstract void listenHold(LucentConnection partyToHold) throws
```

[TsapiInvalidStateException](#), [TsapiMethodNotSupportedException](#),  
[TsapiPrivilegeViolationException](#), [TsapiResourceUnavailableException](#),  
[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.

### **listenUnhold**

```
public abstract void listenUnhold(LucentTerminalConnection partyToUnhold) throws  
TsapiInvalidStateException, TsapiMethodNotSupportedException,  
TsapiPrivilegeViolationException, TsapiResourceUnavailableException,  
TsapiInvalidArgumentException
```

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.

### **listenUnhold**

```
public abstract void listenUnhold(LucentConnection partyToUnhold) throws  
TsapiInvalidStateException, TsapiMethodNotSupportedException,  
TsapiPrivilegeViolationException, TsapiResourceUnavailableException,  
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.

---

---

# Interface com.avaya.jtapi.tsapi.LucentV5TerminalEx

public abstract interface **LucentV5TerminalEx**

extends [LucentV5Terminal](#)

This interface extends `LucentV5Terminal` with additional 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. This interface provides a `removeAgent` method which takes an `int` `reasonCode`, so an application can specify a `reasonCode` when the application removes the agent and in effect logs the agent out.

---

## Method Index

▪ [removeAgent](#)(Agent, int)

This method overrides `AgentTerminal.removeAgent()` to add an additional Lucent DEFINITY G3 PBX Driver Version 5 private data-specific parameter *reasonCode*.

## Methods

● **removeAgent**

```
public abstract void removeAgent(Agent agent,  
                                int reasonCode) throws  
TsapiInvalidArgumentException, TsapiInvalidStateException
```

This method overrides `AgentTerminal.removeAgent()` to add an additional Lucent DEFINITY G3 PBX Driver Version 5 private data-specific parameter *reasonCode*. This is important because when an agent is logged out by removing the agent from the `AgentTerminal`, private data Version 5 provides the ability to specify a `reasonCode`. That capability is exposed with this method.

### Parameters:

`agent` - An `Agent` object representing the association between this `AgentTerminal` and the `ACDAddress` that should be ended by logging out the agent.

`reasonCode` - an `int` conveying to an Application-defined reason code (1-9), or 0 for "no reason".

**Throws:** [TsapiInvalidArgumentException](#)

At least one of the arguments provided is not valid.

**Throws:** [TsapiInvalidStateException](#)

The state of the `AgentTerminal` is not in a state in which it can be logged out of the `ACDAddress`.

---



---

# Interface com.avaya.jtapi.tsapi.LucentV6Agent

public abstract interface **LucentV6Agent**

extends [LucentAgent](#)

This interface extends the LucentAgent interface with features specific to DEFINITY G3 PBX Driver Version 6 private data. When a Provider is bound to a DEFINITY switch which supports V6 private data, this interface may be used to access additional capabilities.

This interface provides access to Lucent *pending state*, *work mode* and *reason code* capabilities.

---

## Method Index

▪ [setState](#)(int, int, int, boolean)

This method overrides LucentAgent.setState to add the Lucent-specific parameters reasonCode and enablePending.

## Methods

● **setState**

```
public abstract boolean setState(int state,
                                int workMode,
                                int reasonCode,
                                boolean enablePending) throws
TsapiInvalidArgumentException, TsapiInvalidStateException
```

This method overrides LucentAgent.setState to add the Lucent-specific parameters reasonCode and enablePending.

### Parameters:

state - The Agent state - now this may be held pending for argument values Agent.WORK\_NOT\_READY and Agent.NOT\_READY (see enablePending).

workMode - The LucentAgent workMode - for pending requests this should be specified as LucentAgent.MODE\_NONE, since workMode may only be specified for Agent state Agent.READY.

reasonCode - An application-defined reasonCode (1-9), which may be specified when the state is set to Agent.NOT\_READY. A zero (0) value is also allowed, meaning "no reason".

enablePending - A boolean flag which, when set to true, specifies that the state, workMode and reason code change may be held pending. The two alternative values have the following implications:

- When enablePending is specified as false, this implies that the application requires that the state, workMode and reasonCode be changed immediately - and that if they *cannot* be applied immediately, such as when the agent is on a call, then the request with this pendingSupport value should **fail** - the JTAPI implementation will throw an exception.
- When enablePending is specified as true, this implies that the application is prepared either for the state, workMode and reasonCode to be changed *immediately*, or, for the changes to be *held pending completion* of the current set of calls active on the agent phone.

### Returns:

true if the requested change in state, workMode and reasonCode is pending; otherwise, the requested change took effect immediately.

**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.

---

---

# Interface com.avaya.jtapi.tsapi.LucentV6Connection

public abstract interface **LucentV6Connection**

extends [LucentV5Connection](#)

This interface extends LucentV5Connection with features specific to DEFINITY G3 PBX Driver Version 6 private data. When a Provider is bound to a DEFINITY switch which supports V6 private data, this interface may be used to access additional capabilities.

The getTrunk method allows an application access to information about the trunk associated with this Connection.

---

## *Method Index*

### ▪ [getTrunk\(\)](#)

If this Connection is associated with a trunk,

## *Methods*

### ● **getTrunk**

```
public abstract javax.telephony.callcenter.CallCenterTrunk getTrunk()
```

If this Connection is associated with a trunk,

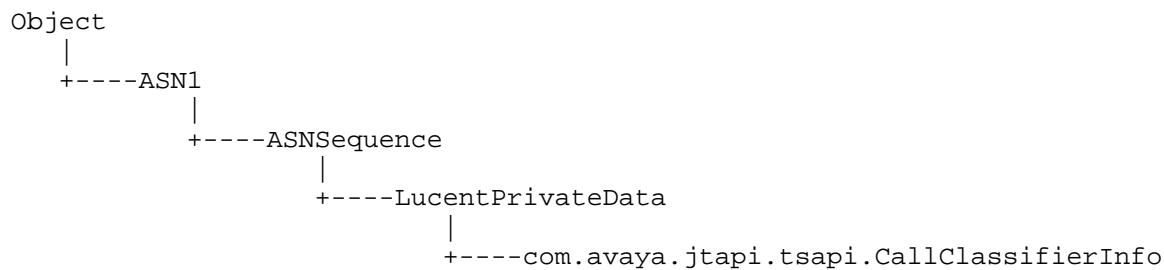
**Returns:**

this method returns the associated CallCenterTrunk. Otherwise returns null.

---

---

# Class com.avaya.jtapi.tsapi.CallClassifierInfo



---

public final class **CallClassifierInfo**

extends LucentPrivateData

Provides information on call classifier port usage.

---

## Variable Index

### ■ [numAvailPorts](#)

The number of available call classifier ports.

### ■ [numInUsePorts](#)

The number of in-use call classifier ports.

## Variables

### ● **numAvailPorts**

```
public int numAvailPorts
```

The number of available call classifier ports.

### ● **numInUsePorts**

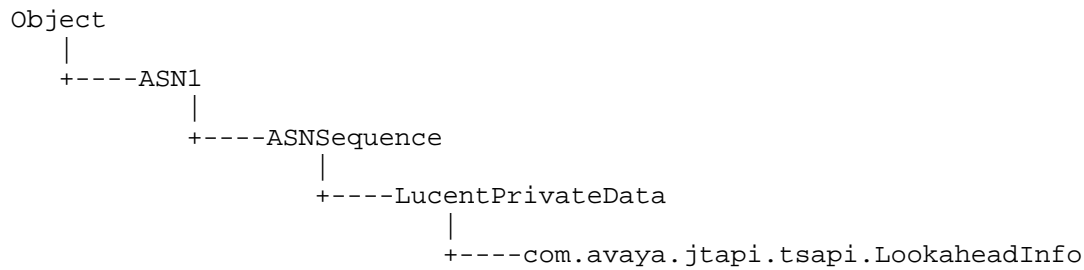
```
public int numInUsePorts
```

The number of in-use call classifier ports.

---

---

# Class com.avaya.jtapi.tsapi.LookaheadInfo



---

public class **LookaheadInfo**

extends LucentPrivateData

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

- [LAI\\_ALL\\_INTERFLOW](#)
- [LAI\\_HIGH](#)
- [LAI\\_LOW](#)
- [LAI\\_MEDIUM](#)
- [LAI\\_NOT\\_IN\\_QUEUE](#)
- [LAI\\_THRESHOLD\\_INTERFLOW](#)
- [LAI\\_TOP](#)
- [LAI\\_VECTORING\\_INTERFLOW](#)

## Method Index

- [getHours\(\)](#)  
Gets the 'hours' part of the event timestamp.
- [getMinutes\(\)](#)  
Gets the 'minutes' part of the event timestamp.
- [getPriority\(\)](#)  
Priority of the interflowed call.
- [getSeconds\(\)](#)

Gets the 'seconds' part of the event timestamp.

#### ▪ [getSourceVDN\(\)](#)

Returns the address of the VDN which overflowed the call.

#### ▪ [getType\(\)](#)

Type of interflow.

## Variables

#### • **LAI\_ALL\_INTERFLOW**

```
public static final short LAI_ALL_INTERFLOW
```

#### • **LAI\_HIGH**

```
public static final short LAI_HIGH
```

#### • **LAI\_LOW**

```
public static final short LAI_LOW
```

#### • **LAI\_MEDIUM**

```
public static final short LAI_MEDIUM
```

#### • **LAI\_NOT\_IN\_QUEUE**

```
public static final short LAI_NOT_IN_QUEUE
```

#### • **LAI\_THRESHOLD\_INTERFLOW**

```
public static final short LAI_THRESHOLD_INTERFLOW
```

#### • **LAI\_TOP**

```
public static final short LAI_TOP
```

#### • **LAI\_VECTORING\_INTERFLOW**

```
public static final short LAI_VECTORING_INTERFLOW
```

## Methods

#### • **getHours**

```
public int getHours()
```

Gets the 'hours' part of the event timestamp.

#### • **getMinutes**

```
public int getMinutes()
```

Gets the 'minutes' part of the event timestamp.

### **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.

### **getSeconds**

```
public int getSeconds()
```

Gets the 'seconds' part of the event timestamp.

### **getSourceVDN**

```
public java.lang.String getSourceVDN()
```

Returns the address of the VDN which overflowed the call.

### **getType**

```
public short getType()
```

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

---

---

# Class com.avaya.jtapi.tsapi.LucentAgentStateInfo

Object  
|  
+----com.avaya.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

### ▪ [state](#)

State of Agent.

### ▪ [workMode](#)

Work Mode for Agent.

## Variables

### ● **state**

public int state

State of Agent.

### ● **workMode**

public int workMode

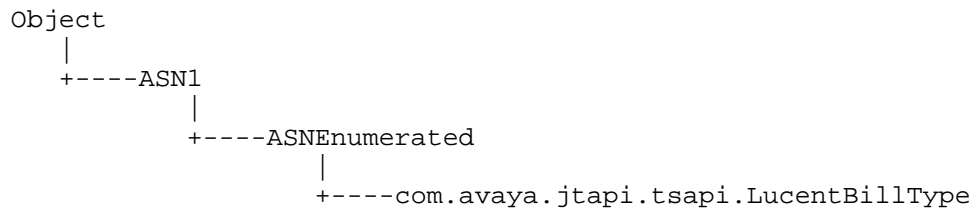
Work Mode for Agent.

---



---

# Class com.avaya.jtapi.tsapi.LucentBillType



---

public final class **LucentBillType**

extends ASNEnumerated

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

See Also:

[LucentV5Call](#)

---

## Variable Index

### ■ [BT\\_FLAT\\_RATE](#)

time independent

### ■ [BT\\_FREE\\_CALL](#)

no charge

### ■ [BT\\_NEW\\_RATE](#)

new rate

### ■ [BT\\_PREMIUM\\_CHARGE](#)

a flat charge in addition to the existing rate

### ■ [BT\\_PREMIUM\\_CREDIT](#)

a flat negative charge in addition to the existing rate

## Variables

### ● **BT\_FLAT\_RATE**

public static final short BT\_FLAT\_RATE

time independent

### ● **BT\_FREE\_CALL**

public static final short BT\_FREE\_CALL

no charge

### ● **BT\_NEW\_RATE**

```
public static final short BT_NEW_RATE  
    new rate
```

### **BT\_PREMIUM\_CHARGE**

```
public static final short BT_PREMIUM_CHARGE  
    a flat charge in addition to the existing rate
```

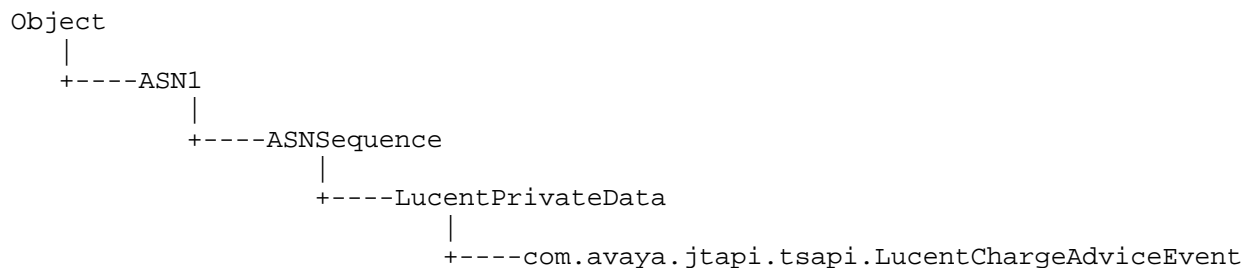
### **BT\_PREMIUM\_CREDIT**

```
public static final short BT_PREMIUM_CREDIT  
    a flat negative charge in addition to the existing rate
```

---

---

# Class com.avaya.jtapi.tsapi.LucentChargeAdviceEvent



---

```
public final class LucentChargeAdviceEvent
extends LucentPrivateData
```

---

## Method Index

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

## Methods

### ▪ getCall

```
public final com.avaya.jtapi.tsapi.LucentCall getCall()
```

The call for which this Charge Advice event is being reported

### ▪ getCalledAddress

```
public final com.avaya.jtapi.tsapi.LucentAddress getCalledAddress()
```

The external address which was dialed

### **getCharge**

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

The number of units charged

### **getChargeError**

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

Charge-related error.

**See Also:**

[LucentChargeError](#)

### **getChargeType**

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

The type of charge being reported

**See Also:**

[LucentChargeType](#)

### **getChargingAddress**

```
public final com.avaya.jtapi.tsapi.LucentAddress getChargingAddress()
```

The address being charged for this call

### **getTrunk**

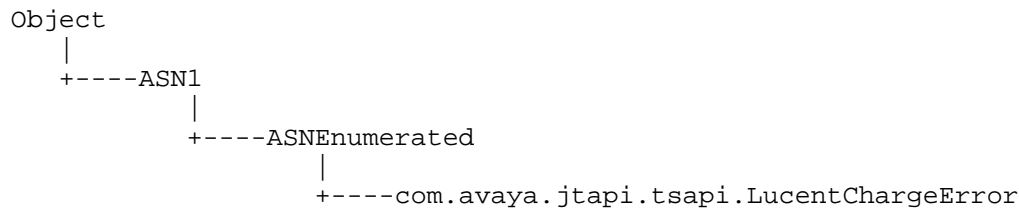
```
public final javax.telephony.callcenter.CallCenterTrunk getTrunk()
```

The trunk reporting the charge

---

---

# Class com.avaya.jtapi.tsapi.LucentChargeError



---

public final class **LucentChargeError**

extends ASNEnumerated

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

See Also:

[LucentChargeAdviceEvent](#)

---

## Variable Index

### ■ [CE\\_CHARGE\\_TOO\\_LARGE](#)

Charge provide by the network is too large

### ■ [CE\\_LESS\\_FINAL\\_CHARGE](#)

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

### ■ [CE\\_NETWORK\\_BUSY](#)

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

### ■ [CE\\_NONE](#)

No error

### ■ [CE\\_NO\\_FINAL\\_CHARGE](#)

Network failed to provide a final charge for the call

## Variables

### ● **CE\_CHARGE\_TOO\_LARGE**

```
public static final short CE_CHARGE_TOO_LARGE
```

Charge provide by the network is too large

### ● **CE\_LESS\_FINAL\_CHARGE**

```
public static final short CE_LESS_FINAL_CHARGE
```

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

### ● **CE\_NETWORK\_BUSY**

```
public static final short CE_NETWORK_BUSY
```

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

 **CE\_NONE**

```
public static final short CE_NONE
```

No error

 **CE\_NO\_FINAL\_CHARGE**

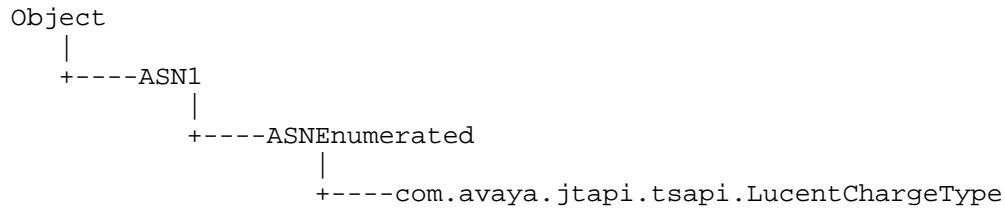
```
public static final short CE_NO_FINAL_CHARGE
```

Network failed to provide a final charge for the call

---

---

# Class com.avaya.jtapi.tsapi.LucentChargeType



---

public final class **LucentChargeType**

extends ASNEnumerated

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

See Also:

[LucentChargeAdviceEvent](#)

---

## Variable Index

### ■ [CT\\_FINAL\\_CHARGE](#)

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

### ■ [CT\\_INTERMEDIATE\\_CHARGE](#)

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

### ■ [CT\\_SPLIT\\_CHARGE](#)

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

## Variables

### ● **CT\_FINAL\_CHARGE**

```
public static final short CT_FINAL_CHARGE
```

This charge is send 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.

### ● **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.

### ● **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

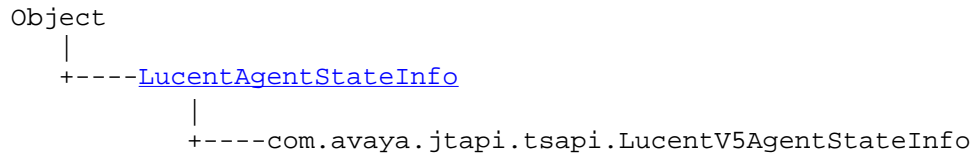
station and transferred to a second station, and if CDR call splitting is enabled, then CDR and the Charge Advice Events would charge the first station up to the time of the transfer, and second station after that. A split charge reflects the charge for the call up to the time the split charge is sent (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.avaya.jtapi.tsapi.LucentV5AgentStateInfo



public 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

### ■ [reasonCode](#)

Application-defined reason code (1-9)

## Variables

### ● [reasonCode](#)

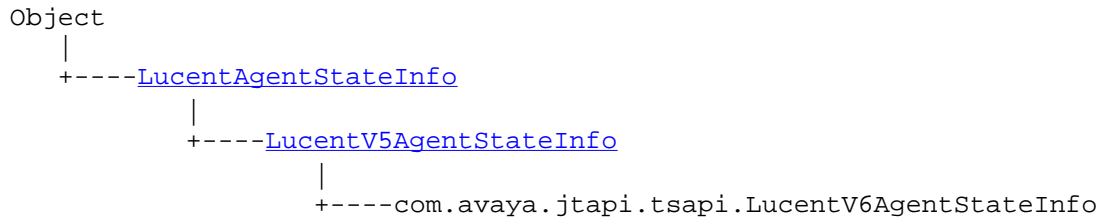
public int reasonCode

Application-defined reason code (1-9)

---

---

# Class com.avaya.jtapi.tsapi.LucentV6AgentStateInfo



public class **LucentV6AgentStateInfo**

extends [LucentV5AgentStateInfo](#)

This is the object that is returned by the query getStateInfo method of the LucentAgent interface, with the DEFINITY G3 Driver supporting and using Private data version 6.

An Agent implementing the LucentAgent interface has an Agent state, an LucentAgent workMode and a LucentV5Agent reasonCode. In prior releases, the getStateInfo method returned this object, which carried that status information (state, workMode, reasonCode) about an Agent; with private data version 6, this method will also return any *pending* status information for the Agent.

Status information (state, workMode and reasonCode) may be held pending, to be applied when all calls currently associated with the Agent are completed, as part of a new DEFINITY G3 feature supported by private data version 6.

See Also:

[LucentV6Agent](#)

---

## Variable Index

### ▪ [pendingReasonCode](#)

application-defined reason code held pending, to be applied after the current call (or held call) has ended at the agent station.

### ▪ [pendingState](#)

state held pending, to be applied after the current call (or held call) has ended at the agent station.

## Variables

### ▪ **pendingReasonCode**

public int pendingReasonCode

application-defined reason code held pending, to be applied after the current call (or held call) has ended at the agent station. This will be non-zero only when pendingState == Agent.NOT\_READY.

See Also:

[reasonCode](#)

### ▪ **pendingState**

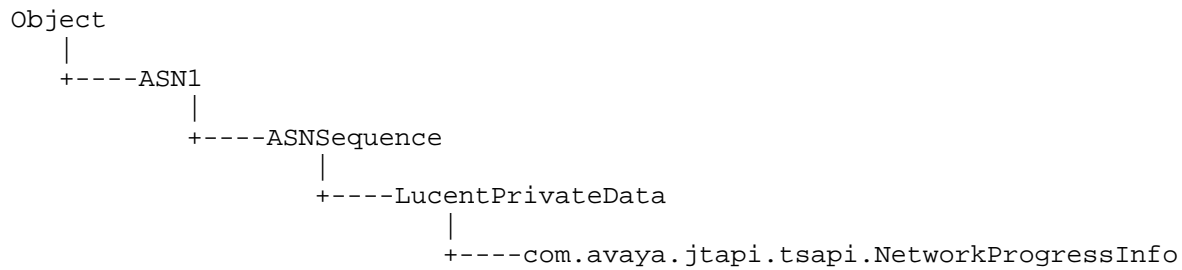
public int pendingState

state held pending, to be applied after the current call (or held call) has ended at the agent station. Possible returned values are Agent.UNKNOWN (if no state is held pending), Agent.NOT\_READY and Agent.WORK\_NOT\_READY.

---

---

# Class com.avaya.jtapi.tsapi.NetworkProgressInfo



public class **NetworkProgressInfo**

extends LucentPrivateData

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

---

## Variable Index

■ [PD\\_CALL\\_OFF\\_ISDN](#)

■ [PD\\_CALL\\_ON\\_ISDN](#)

■ [PD\\_DEST\\_NOT\\_ISDN](#)

■ [PD\\_INBAND](#)

■ [PD\\_ORIG\\_NOT\\_ISDN](#)

■ [PL\\_PRIV\\_REMOTE](#)

■ [PL\\_PUB\\_LOCAL](#)

■ [PL\\_PUB\\_REMOTE](#)

■ [PL\\_USER](#)

■ [progressDescription](#)

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

■ [progressLocation](#)

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

## Variables

● [PD\\_CALL\\_OFF\\_ISDN](#)

```
public static final short PD_CALL_OFF_ISDN
```

● [PD\\_CALL\\_ON\\_ISDN](#)

```
public static final short PD_CALL_ON_ISDN
```

## **PD\_DEST\_NOT\_ISDN**

```
public static final short PD_DEST_NOT_ISDN
```

## **PD\_INBAND**

```
public static final short PD_INBAND
```

## **PD\_ORIG\_NOT\_ISDN**

```
public static final short PD_ORIG_NOT_ISDN
```

## **PL\_PRIV\_REMOTE**

```
public static final short PL_PRIV_REMOTE
```

## **PL\_PUB\_LOCAL**

```
public static final short PL_PUB_LOCAL
```

## **PL\_PUB\_REMOTE**

```
public static final short PL_PUB_REMOTE
```

## **PL\_USER**

```
public static final short PL_USER
```

## **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

## **progressLocation**

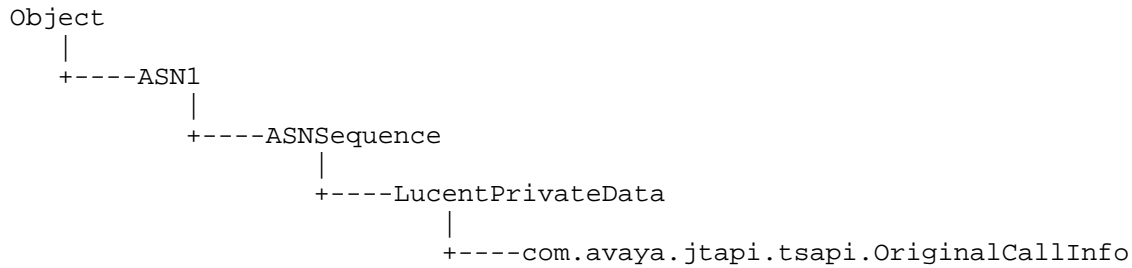
```
public short progressLocation
```

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

---

---

# Class com.avaya.jtapi.tsapi.OriginalCallInfo



---

public class **OriginalCallInfo**

extends LucentPrivateData

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

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

**See Also:**

[LucentCallInfo](#)

---

## Variable Index

- [OR\\_CONFERENCED](#)
- [OR\\_CONSULTATION](#)
- [OR\\_NEW\\_CALL](#)
- [OR\\_NONE](#)
- [OR\\_TRANSFERRED](#)

## Method Index

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

## ▪ [getUserEnteredCode\(\)](#)

Get the original collected digits for this call.

## ▪ [getUserToUserInfo\(\)](#)

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

# Variables

## • **OR\_CONFERENCED**

```
public static final short OR_CONFERENCED
```

## • **OR\_CONSULTATION**

```
public static final short OR_CONSULTATION
```

## • **OR\_NEW\_CALL**

```
public static final short OR_NEW_CALL
```

## • **OR\_NONE**

```
public static final short OR_NONE
```

## • **OR\_TRANSFERRED**

```
public static final short OR_TRANSFERRED
```

# Methods

## • **getCalledDevice**

```
public javax.telephony.Address getCalledDevice()
```

Get the original called device for this call.

## • **getCallingDevice**

```
public javax.telephony.Address getCallingDevice()
```

Get the original calling device for this call.

## • **getLookaheadInfo**

```
public com.avaya.jtapi.tsapi.LookaheadInfo getLookaheadInfo()
```

Get the original lookahead information for this call.

## • **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.

## • **getTrunk**

```
public javax.telephony.callcenter.CallCenterTrunk getTrunk()
```

Get the original trunk device for this call.

### **getUserEnteredCode**

```
public com.avaya.jtapi.tsapi.UserEnteredCode getUserEnteredCode()
```

Get the original collected digits for this call.

### **getUserToUserInfo**

```
public com.avaya.jtapi.tsapi.UserToUserInfo getUserToUserInfo()
```

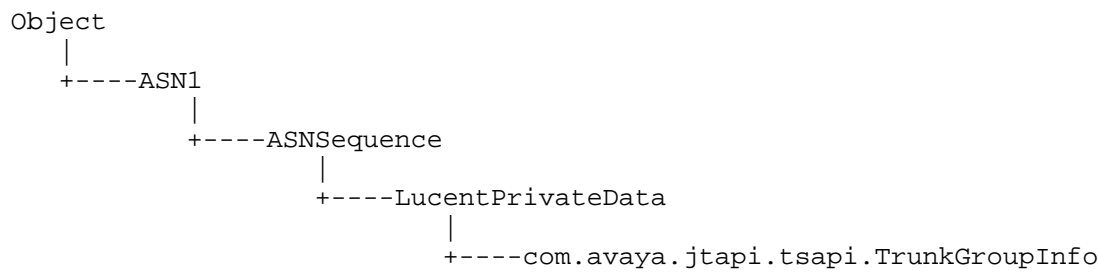
Get the original user-to-user information for this call.

---



---

# Class com.avaya.jtapi.tsapi.TrunkGroupInfo



---

public final class **TrunkGroupInfo**

extends LucentPrivateData

Provides information on trunk group usage.

---

## Variable Index

### ■ [idleTrunks](#)

The number of idle trunks.

### ■ [usedTrunks](#)

The number of in-use trunks.

## Variables

### ● **idleTrunks**

```
public int idleTrunks
```

The number of idle trunks.

### ● **usedTrunks**

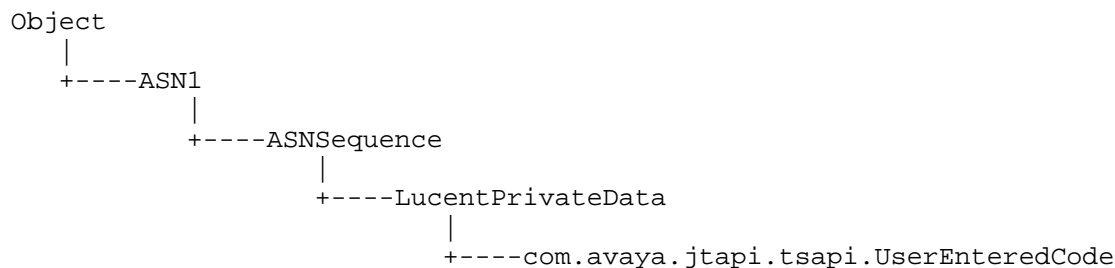
```
public int usedTrunks
```

The number of in-use trunks.

---

---

# Class com.avaya.jtapi.tsapi.UserEnteredCode



---

public final class **UserEnteredCode**

extends LucentPrivateData

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

- [UE\\_ANY](#)
- [UE\\_CALL\\_PROMPTER](#)
- [UE\\_COLLECT](#)
- [UE\\_DATA\\_BASE\\_PROVIDED](#)
- [UE\\_ENTERED](#)
- [UE\\_LOGIN\\_DIGITS](#)
- [UE\\_TONE\\_DETECTOR](#)

# Method Index

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

# Variables

## • UE\_ANY

```
public static final short UE_ANY
```

## • UE\_CALL\_PROMPTER

```
public static final short UE_CALL_PROMPTER
```

## • UE\_COLLECT

```
public static final short UE_COLLECT
```

## • UE\_DATA\_BASE\_PROVIDED

```
public static final short UE_DATA_BASE_PROVIDED
```

## • UE\_ENTERED

```
public static final short UE_ENTERED
```

## • UE\_LOGIN\_DIGITS

```
public static final short UE_LOGIN_DIGITS
```

## UE\_TONE\_DETECTOR

```
public static final short UE_TONE_DETECTOR
```

# Methods

## getCollectVDN

```
public javax.telephony.callcenter.ACDManagerAddress getCollectVDN()
```

Returns the ACDManagerAddress of the VDN which collected the digits

## getDigits

```
public java.lang.String getDigits()
```

Returns the collected digits

## getIndicator

```
public short getIndicator()
```

Returns UE\_COLLECT or UE\_ENTERED

## getType

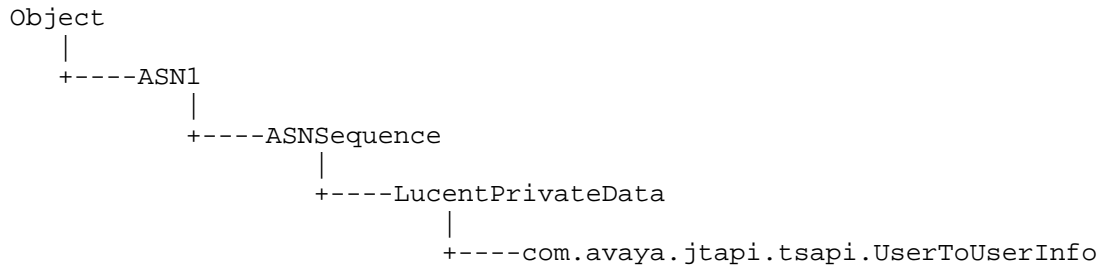
```
public short getType()
```

Returns the type of digits collected

---

---

# Class com.avaya.jtapi.tsapi.UserToUserInfo



---

public final class **UserToUserInfo**

extends LucentPrivateData

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

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

**See Also:**

[LucentCallInfo](#)

---

## Constructor Index

■ [com.avaya.jtapi.tsapi.UserToUserInfo\(String\)](#)

construct a UserToUserInfo object from an ASCII string

■ [com.avaya.jtapi.tsapi.UserToUserInfo\(byte\[\]\)](#)

construct a UserToUserInfo object from a byte array

## Method Index

■ [getBytes\(\)](#)

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

■ [getString\(\)](#)

return user-to-user info as an ASCII string

■ [isAscii\(\)](#)

query whether sender encoded UUI as ASCII or binary

## Constructors

## **UserToUserInfo**

```
public UserToUserInfo(String _data)
    construct a UserToUserInfo object from an ASCII string
```

## **UserToUserInfo**

```
public UserToUserInfo(byte[] _data)
    construct a UserToUserInfo object from a byte array
```

# Methods

## **getBytes**

```
public byte[] getBytes()
    return user-to-user info as a (binary) byte array
```

## **getString**

```
public java.lang.String getString()
    return user-to-user info as an ASCII string
```

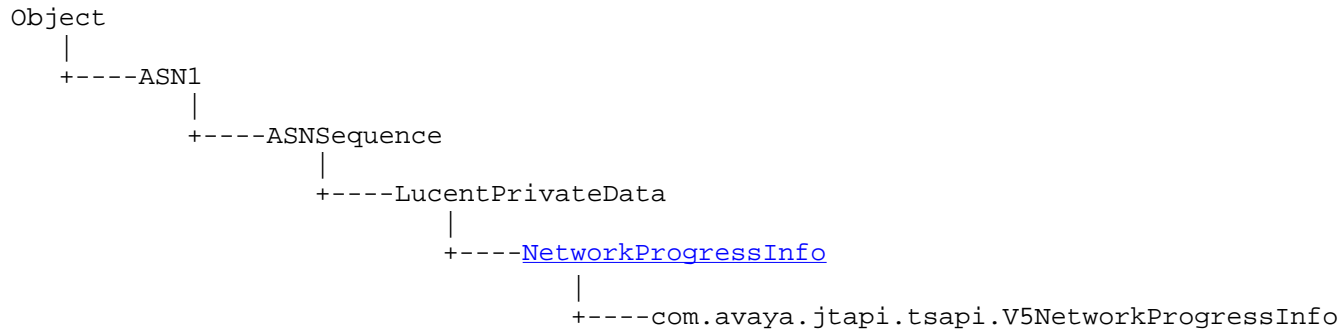
## **isAscii**

```
public boolean isAscii()
    query whether sender encoded UUI as ASCII or binary
```

---

---

# Class com.avaya.jtapi.tsapi.V5NetworkProgressInfo



public final class **V5NetworkProgressInfo**

extends [NetworkProgressInfo](#)

Adds DEFINITY G3V6-specific data to the NetworkProgressInfo event

---

## Variable Index

▪ [trunk](#)

## Variables

▪ [trunk](#)

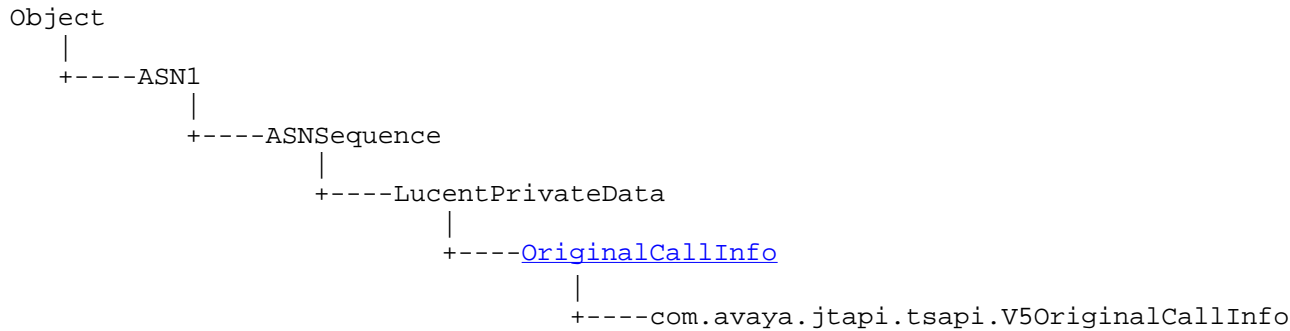
```
public com.avaya.jtapi.tsapi.TsapiTrunk trunk
```

---



---

# Class com.avaya.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

### ▪ [canSetBillRate\(\)](#)

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

### ▪ [getCallOriginatorType\(\)](#)

Get the originator type for this call, such as coin call, 800 service call, or cellular call.

### ▪ [getUCID\(\)](#)

Get the Universal Call ID for this call.

### ▪ [hasCallOriginatorType\(\)](#)

Query whether CallOriginatorType is available for this call.

## Methods

### • canSetBillRate

```
public boolean canSetBillRate()
```

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

### • 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

### **getUCID**

```
public java.lang.String getUCID()  
    Get the Universal Call ID for this call. (Requires DEFINITY G3V6.)
```

### **hasCallOriginatorType**

```
public boolean hasCallOriginatorType()  
    Query whether CallOriginatorType is available for this call.
```

---

# Chapter 4

## Using Telephony Services Private Data Extensions

This page 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.

Additionally, this chapter describes 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 described in "[Using Telephony Services DEFINITY-Specific Extensions](#).) It is assumed that the reader is familiar with the Java programming language, JTAPI, Avaya Telephony Services Application Programmer's Interface (TSAPI) and its private data mechanism.

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 "[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 "[Using Telephony Services Extensions to JTAPI](#)."

If you are an application programmer who is using JTAPI to develop applications for the DEFINITY switch, ignore this chapter and refer to "[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 "[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.

### 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()

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.

### 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.avaya.jtapi.tsapi

## Interface Index

- [ITsapiCallIDPrivate](#)
- [ITsapiConnIDPrivate](#)
- [ITsapiPeer](#)
- [ITsapiProvider](#)
- [ITsapiProviderPrivate](#)
- [ITsapiProviderTsapiInServiceEvent](#)
- [ITsapiProviderTsapiInitializingEvent](#)
- [ITsapiProviderTsapiOutOfServiceEvent](#)
- [ITsapiProviderTsapiShutdownEvent](#)
- [ITsapiRoutePrivate](#)

## Class Index

- [ConnectionID](#)
- [ExtendedDeviceID](#)
- [TsapiPeer](#)
- [TsapiPrivate](#)

## Exception Index

---

# Interface com.avaya.jtapi.tsapi.ITsapiCallIDPrivate

public abstract interface **ITsapiCallIDPrivate**

ITsapiCallIDPrivate lets you retrieve TSAPI information associated with a JTAPI Call.

---

## *Method Index*

### ▪ [getTsapiCallID\(\)](#)

Retrieves the TSAPI CallID associated with a JTAPI Call.

## *Methods*

### • **getTsapiCallID**

public abstract int getTsapiCallID()

Retrieves the TSAPI CallID associated with a JTAPI Call.

---

---

# Interface com.avaya.jtapi.tsapi.ITsapiConnIDPrivate

public abstract interface **ITsapiConnIDPrivate**

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

**See Also:**

[ConnectionID](#)

---

## *Method Index*

▪ [getTsapiConnectionID\(\)](#)

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

## *Methods*

● **getTsapiConnectionID**

public abstract com.avaya.jtapi.tsapi.ConnectionID getTsapiConnectionID()

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

**See Also:**

[ConnectionID](#)

---



---

# Interface com.avaya.jtapi.tsapi.ITsapiPeer

public abstract 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*

▪ [addVendor](#)(String, String)

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

## *Methods*

▪ **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.private.data.

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

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

---

---

# Interface com.avaya.jtapi.tsapi.ITsapiProvider

public abstract interface **ITsapiProvider**

extends [Provider](#), [CallCenterProvider](#)

ITsapiProvider adds methods to obtain vendor-specific version information.

See Also:

[addVendor](#)

---

## Variable Index

### ■ [TSAPI\\_INITIALIZING](#)

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

### ■ [TSAPI\\_IN\\_SERVICE](#)

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

### ■ [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.

### ■ [TSAPI\\_SHUTDOWN](#)

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

## Method Index

### ■ [getTsapiState\(\)](#)

Returns the TSAPI state of the provider.

### ■ [getVendor\(\)](#)

Returns the data vendor name.

### ■ [getVendorVersion\(\)](#)

Returns the negotiated vendor data version.

### ■ [setDebugPrinting\(boolean\)](#)

Enable/disable debug printing in the debug build version

### ■ [updateAddresses\(\)](#)

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

## Variables

### ● [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`.

## ● 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.

## ● 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.

## ● TSAPI\_SHUTDOWN

```
public static final int TSAPI_SHUTDOWN
```

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

# Methods

## ● `getTsapiState`

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

Returns the TSAPI state of the provider.

## ● `getVendor`

```
public abstract java.lang.String getVendor()
```

Returns the data vendor name.

## ● `getVendorVersion`

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

Returns the negotiated vendor data version.

## ● `setDebugPrinting`

```
public abstract void setDebugPrinting(boolean enable)
```

Enable/disable debug printing in the debug build version

## ● `updateAddresses`

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

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

---

---

# Interface com.avaya.jtapi.tsapi.ITsapiProviderPrivate

public abstract interface **ITsapiProviderPrivate**

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

See Also:

[ConnectionID](#), [ExtendedDeviceID](#)

---

## Method Index

▪ [getAddress](#)(ExtendedDeviceID)

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

▪ [getCall](#)(int)

Returns a JTAPI Call associated with a TSAPI Call ID.

▪ [getConnection](#)(ConnectionID, Address)

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

▪ [getTerminal](#)(ExtendedDeviceID)

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

▪ [getTerminalConnection](#)(ConnectionID, Terminal)

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

## Methods

● **getAddress**

```
public abstract javax.telephony.Address getAddress(ExtendedDeviceID deviceID)
```

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

**Parameters:**

deviceID - The Extended Device ID.

**See Also:**

[ExtendedDeviceID](#)

● **getCall**

```
public abstract javax.telephony.Call getCall(int callID)
```

Returns a JTAPI Call associated with a TSAPI Call ID.

**Parameters:**

callID - The Call ID.

● **getConnection**

```
public abstract javax.telephony.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](#)

 **getTerminal**

```
public abstract javax.telephony.Terminal getTerminal(ExtendedDeviceID deviceID)
```

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

**Parameters:**

deviceID - The Extended Device ID.

**See Also:**

[ExtendedDeviceID](#)

 **getTerminalConnection**

```
public abstract javax.telephony.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.avaya.jtapi.tsapi.ITsapiProviderTsapiInServiceEvent

public abstract interface **ITsapiProviderTsapiInServiceEvent**

This interface indicates the provider is in the Tsapi in service state.

---

## *Method Index*

### ▪ [getTsapiState\(\)](#)

Returns the Tsapi state associated with this event, `ITsapiProvider.TSAPI_IN_SERVICE<\CODE>`.

## *Methods*

### ▪ **getTsapiState**

public abstract int **getTsapiState()**

Returns the Tsapi state associated with this event, `ITsapiProvider.TSAPI_IN_SERVICE<\CODE>`.

---

---

# Interface

## com.avaya.jtapi.tsapi.ITsapiProviderTsapiInitializingEvent

public abstract interface **ITsapiProviderTsapiInitializingEvent**

This interface indicates the provider is in the Tsapi initializing state.

---

## *Method Index*

### ▪ [getTsapiState\(\)](#)

Returns the Tsapi state associated with this event, `ITsapiProvider.INITIALIZING`.

## *Methods*

### ▪ **getTsapiState**

public abstract int `getTsapiState()`

Returns the Tsapi state associated with this event, `ITsapiProvider.INITIALIZING`.

---

---

# Interface

## com.avaya.jtapi.tsapi.ITsapiProviderTsapiOutOfServiceEvent

public abstract interface **ITsapiProviderTsapiOutOfServiceEvent**

This interface indicates the provider is in the Tsapi out of service state.

---

## Method Index

### ▪ [getTsapiState\(\)](#)

Returns the Tsapi state associated with this event, `ITsapiProvider.TSAPI_OUT_OF_SERVICE<\CODE>`.

## Methods

### ▪ `getTsapiState`

public abstract int `getTsapiState()`

Returns the Tsapi state associated with this event, `ITsapiProvider.TSAPI_OUT_OF_SERVICE<\CODE>`.

---



---

# Interface

## com.avaya.jtapi.tsapi.ITsapiProviderTsapiShutdownEvent

public abstract interface **ITsapiProviderTsapiShutdownEvent**

This interface indicates the provider is in the Tsapi shutdown state.

---

## *Method Index*

### ▪ [getTsapiState\(\)](#)

Returns the Tsapi state associated with this event, `ITsapiProvider.TSAPI_SHUTDOWN<\CODE>`.

## *Methods*

### ▪ **getTsapiState**

public abstract int **getTsapiState()**

Returns the Tsapi state associated with this event, `ITsapiProvider.TSAPI_SHUTDOWN<\CODE>`.

---

---

# Interface com.avaya.jtapi.tsapi.ITsapiRoutePrivate

public abstract interface **ITsapiRoutePrivate**

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

---

## *Method Index*

### ▪ [getRouteCrossRefID\(\)](#)

Retrieves the TSAPI RouteCrossReferenceID associated with a JTAPI Route Session.

### ▪ [getRouteRegisterID\(\)](#)

Retrieves the TSAPI RouteRegisterID with a JTAPI Route Session.

## *Methods*

### ● **getRouteCrossRefID**

public abstract int getRouteCrossRefID()

Retrieves the TSAPI RouteCrossReferenceID associated with a JTAPI Route Session.

### ● **getRouteRegisterID**

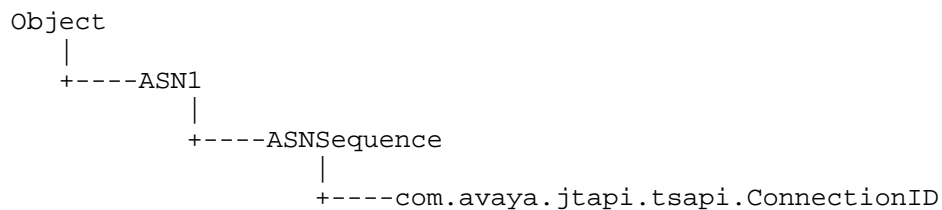
public abstract int getRouteRegisterID()

Retrieves the TSAPI RouteRegisterID with a JTAPI Route Session.

---

---

# Class com.avaya.jtapi.tsapi.ConnectionID



---

public final class **ConnectionID**

extends ASNSequence

This class exposes the TSAPI ConnectionID, for implementors of other vendors' private data.

---

## Variable Index

▪ [DYNAMIC\\_ID](#)

▪ [STATIC\\_ID](#)

## Method Index

▪ [equals](#)(Object)

▪ [hashCode](#)()

▪ [toString](#)()

## Variables

• **DYNAMIC\_ID**

public static final short DYNAMIC\_ID

• **STATIC\_ID**

public static final short STATIC\_ID

## Methods

• **equals**

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

**Overrides:**

equals in class Object

 **hashCode**

```
public int hashCode()
```

**Overrides:**

hashCode in class Object

 **toString**

```
public java.lang.String toString()
```

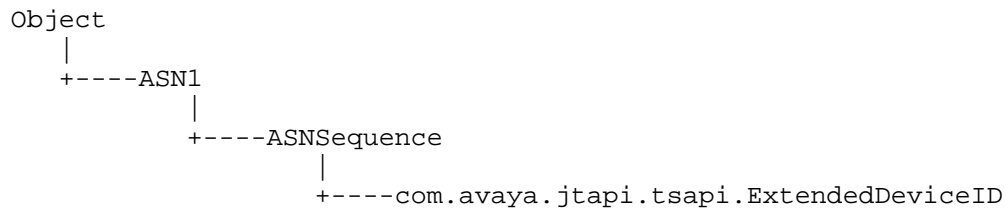
**Overrides:**

toString in class Object

---

---

# Class com.avaya.jtapi.tsapi.ExtendedDeviceID



public final class **ExtendedDeviceID**

extends ASNSequence

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

### ■ [DEVICE\\_IDENTIFIER](#)

Device ID Type.

### ■ [EXPLICIT\\_PRIVATE\\_ABBREVIATED](#)

Device ID Type.

### ■ [EXPLICIT\\_PRIVATE\\_LEVEL1\\_REGIONAL\\_NUMBER](#)

Device ID Type.

### ■ [EXPLICIT\\_PRIVATE\\_LEVEL2\\_REGIONAL\\_NUMBER](#)

Device ID Type.

### ■ [EXPLICIT\\_PRIVATE\\_LEVEL3\\_REGIONAL\\_NUMBER](#)

Device ID Type.

### ■ [EXPLICIT\\_PRIVATE\\_LOCAL\\_NUMBER](#)

Device ID Type.

### ■ [EXPLICIT\\_PRIVATE\\_PTN\\_SPECIFIC\\_NUMBER](#)

Device ID Type.

### ■ [EXPLICIT\\_PRIVATE\\_UNKNOWN](#)

Device ID Type.

### ■ [EXPLICIT\\_PUBLIC\\_ABBREVIATED](#)

Device ID Type.

### ■ [EXPLICIT\\_PUBLIC\\_INTERNATIONAL](#)

Device ID Type.

### ■ [EXPLICIT\\_PUBLIC\\_NATIONAL](#)

Device ID Type.

## ▪ [EXPLICIT\\_PUBLIC\\_NETWORK\\_SPECIFIC](#)

Device ID Type.

## ▪ [EXPLICIT\\_PUBLIC\\_SUBSCRIBER](#)

Device ID Type.

## ▪ [EXPLICIT\\_PUBLIC\\_UNKNOWN](#)

Device ID Type.

## ▪ [ID\\_NOT\\_KNOWN](#)

Device ID Status of *ID\_NOT\_KNOWN* indicates the Device ID is not known.

## ▪ [ID\\_NOT\\_REQUIRED](#)

Device ID Status of *ID\_NOT\_REQUIRED* indicates the Device ID is not required.

## ▪ [ID\\_PROVIDED](#)

Device ID Status of *ID\_PROVIDED* indicates the Device ID is valid

## ▪ [IMPLICIT\\_PRIVATE](#)

Device ID Type.

## ▪ [IMPLICIT\\_PUBLIC](#)

Device ID Type.

## ▪ [OTHER\\_PLAN](#)

Device ID Type.

## ▪ [TRUNK\\_GROUP\\_IDENTIFIER](#)

Device ID Type.

## ▪ [TRUNK\\_IDENTIFIER](#)

Device ID Type.

# Constructor Index

## ▪ [com.avaya.jtapi.tsapi.ExtendedDeviceID](#)(String, short, short)

Construct an ExtendedDeviceID.

# Method Index

## ▪ [toString\(\)](#)

# Variables

## • [DEVICE\\_IDENTIFIER](#)

```
public static final short DEVICE_IDENTIFIER
```

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

## • [EXPLICIT\\_PRIVATE\\_ABBREVIATED](#)

public static final short EXPLICIT\_PRIVATE\_ABBREVIATED

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

**EXPLICIT\_PRIVATE\_LEVEL1\_REGIONAL\_NUMBER**

public static final short EXPLICIT\_PRIVATE\_LEVEL1\_REGIONAL\_NUMBER

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

**EXPLICIT\_PRIVATE\_LEVEL2\_REGIONAL\_NUMBER**

public static final short EXPLICIT\_PRIVATE\_LEVEL2\_REGIONAL\_NUMBER

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

**EXPLICIT\_PRIVATE\_LEVEL3\_REGIONAL\_NUMBER**

public static final short EXPLICIT\_PRIVATE\_LEVEL3\_REGIONAL\_NUMBER

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

**EXPLICIT\_PRIVATE\_LOCAL\_NUMBER**

public static final short EXPLICIT\_PRIVATE\_LOCAL\_NUMBER

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

**EXPLICIT\_PRIVATE\_PTN\_SPECIFIC\_NUMBER**

public static final short EXPLICIT\_PRIVATE\_PTN\_SPECIFIC\_NUMBER

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

**EXPLICIT\_PRIVATE\_UNKNOWN**

public static final short EXPLICIT\_PRIVATE\_UNKNOWN

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

**EXPLICIT\_PUBLIC\_ABBREVIATED**

public static final short EXPLICIT\_PUBLIC\_ABBREVIATED

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

**EXPLICIT\_PUBLIC\_INTERNATIONAL**

public static final short EXPLICIT\_PUBLIC\_INTERNATIONAL

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

**EXPLICIT\_PUBLIC\_NATIONAL**

public static final short EXPLICIT\_PUBLIC\_NATIONAL

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

**EXPLICIT\_PUBLIC\_NETWORK\_SPECIFIC**

public static final short EXPLICIT\_PUBLIC\_NETWORK\_SPECIFIC

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

**EXPLICIT\_PUBLIC\_SUBSCRIBER**

public static final short EXPLICIT\_PUBLIC\_SUBSCRIBER

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

## ● EXPLICIT\_PUBLIC\_UNKNOWN

```
public static final short EXPLICIT_PUBLIC_UNKNOWN
```

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

## ● 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.

## ● 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.

## ● ID\_PROVIDED

```
public static final short ID_PROVIDED
```

Device ID Status of *ID\_PROVIDED* indicates the Device ID is valid

## ● IMPLICIT\_PRIVATE

```
public static final short IMPLICIT_PRIVATE
```

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

## ● IMPLICIT\_PUBLIC

```
public static final short IMPLICIT_PUBLIC
```

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

## ● OTHER\_PLAN

```
public static final short OTHER_PLAN
```

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

## ● TRUNK\_GROUP\_IDENTIFIER

```
public static final short TRUNK_GROUP_IDENTIFIER
```

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

## ● TRUNK\_IDENTIFIER

```
public static final short TRUNK_IDENTIFIER
```

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

# CONSTRUCTORS

## ● 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

**toString**

```
public java.lang.String toString()
```

**Overrides:**

toString in class Object

---

---

# Class com.avaya.jtapi.tsapi.TsapiPeer

Object  
|  
+----com.avaya.jtapi.tsapi.TsapiPeer

---

public class **TsapiPeer**  
extends Object  
implements [ITsapiPeer](#)  
TsapiPeer implements JtapiPeer.

---

## Variable Index

### ■ [VERSION](#)

GA Version.

## Constructor Index

### ■ [com.avaya.jtapi.tsapi.TsapiPeer\(\)](#)

This constructor is used by JtapiPeerFactory to create an instance of the peer.

## Method Index

■ [addVendor](#)(String, String)

■ [getName](#)()

■ [getProvider](#)(String)

■ [getServices](#)()

## Variables

### ● [VERSION](#)

public static final java.lang.String VERSION  
GA Version.

# Constructors

## ● TsapiPeer

```
public TsapiPeer()
```

This constructor is used by JtapiPeerFactory to create an instance of the peer.

# Methods

## ● addVendor

```
public final void addVendor(String name,  
                             String versions)
```

## ● getName

```
public java.lang.String getName()
```

## ● getProvider

```
public final javax.telephony.Provider getProvider(String providerString)
```

## ● getServices

```
public final java.lang.String[] getServices()
```

---

---

# Class com.avaya.jtapi.tsapi.TsapiPrivate

Object  
|  
+----com.avaya.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

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

## Constructor Index

- [com.avaya.jtapi.tsapi.TsapiPrivate](#)(byte[])  
Construct a TSAPI data object.
- [com.avaya.jtapi.tsapi.TsapiPrivate](#)(byte[], boolean)  
Construct a TSAPI data object.


## Method Index

- [getData\(\)](#)  
Return the byte array containing the raw data.

## Variables

- [data](#)

```
public byte[] data
```

 **tsType**

```
public int tsType
```

 **vendor**

```
public java.lang.String vendor
```

# Constructors

 **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()*).

 **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

 **getData**

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

Return the byte array containing the raw data.

---

# Appendix A

## More About Private Data

The information on the JTAPI for Private Data page describes the level of support the CentreVu Computer Telephony (CVCT) 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. (For 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 see [Using Telephony Services DEFINITY-Specific Extensions](#).)

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 "[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, "[Using Telephony Services Extensions to JTAPI](#)."

If you are an application programmer who is using JTAPI to develop applications for the DEFINITY switch, ignore this appendix and refer to "[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 "[Telephony Services DEFINITY-Specific Extensions to JTAPI](#)."

- [Telephony Services Implementation of JTAPI for Private Data](#)
  - [TSAPI Requests with Associated JTAPI Interfaces and Methods](#)
  - [TSAPI Requests without Associated JTAPI Mapping](#)
  - [Mapping of Possible JTAPI Events to TSAPI Events](#)
- 

## Telephony Services Implementation of JTAPI for Private Data

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

The Avaya 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 [TSAPI Requests with Associated JTAPI Interfaces and Methods](#). See [TSAPI Requests without Associated JTAPI Mapping](#) for 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.private.data.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.

[Mapping of Possible JTAPI Events to TSAPI Events](#) 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

TSAPI Requests	JTAPI Interfaces	JTAPI Methods
<code>cstaMakeCall</code>	<code>Call</code>	<code>connect</code>
<code>cstaClearConnection</code>	<code>Connection</code>	<code>disconnect</code>
<code>acsEnumServerNames</code>	<code>JtapiPeer</code>	<code>getServices</code>
<code>acsOpenStream</code>	<code>JtapiPeer</code>	<code>getProvider</code>
<code>acsCloseStream</code>	<code>Provider</code>	<code>shutdown</code>
<code>cstaAnswerCall</code>	<code>TerminalConnection</code>	<code>answer</code>
<code>cstaSetAgentState</code>	<code>AgentTerminal</code> <code>Agent</code>	<code>addAgent</code> <code>setState</code>
<code>cstaQueryAgentState</code>	<code>Agent</code>	<code>getState</code>
<code>cstaMakePredictiveCall</code>	<code>CallCenterCall</code>	<code>connectPredictive</code>
<code>cstaRouteRegisterReq</code>	<code>RouteAddress</code>	<code>registerRouteCallback</code>
<code>cstaRouteRegisterCancel</code>	<code>RouteAddress</code>	<code>cancelRouteCallback</code>
<code>cstaRouteSelectInv</code>	<code>RouteSession</code>	<code>selectRoute</code>
<code>cstaRouteEndInv</code>	<code>RouteSession</code>	<code>endRoute</code>
<code>cstaSetForwarding</code>	<code>CallControlAddress</code>	<code>setForwarding</code> <code>cancelForwarding</code>
<code>cstaQueryForwarding</code>	<code>CallControlAddress</code>	<code>getForwarding</code>
<code>cstaQueryDoNotDisturb</code>	<code>CallControlAddress</code>	<code>getDoNotDisturb</code>
<code>cstaSetDoNotDisturb</code>	<code>CallControlAddress</code>	<code>setDoNotDisturb</code>
<code>cstaQueryMsgWaitingInd</code>	<code>CallControlAddress</code>	<code>getMessageWaiting</code>
<code>cstaSetMsgWaitingInd</code>	<code>CallControlAddress</code>	<code>setMessageWaiting</code>

cstaClearCall	CallControlCall	drop
cstaConferenceCall	CallControlCall	conference
cstaTransferCall	CallControlCall	transfer
cstaConsultationCall	CallControlCall	consult
cstaDeflectCall	CallControlConnection	redirect
cstaQueryDoNotDisturb	CallControlTerminal	getDoNotDisturb
cstaSetDnd	CallControlTerminal	setDoNotDisturb
cstaPickupCall	CallControlTerminal	pickup
cstaGroupPickupCall	CallControlTerminal	pickupFromGroup
cstaHoldCall	CallControlTerminalConnection	hold
cstaRetrieveCall	CallControlTerminalConnection	unhold
cstaSendPrivateEvent	Private Data	sendPrivateData

### TSAPI Requests without Associated JTAPI Mapping

Here is a list of 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.

<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**

Here is a list of 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.

<b>TSAPI Events</b>	<b>Possible JTAPI Event in Array with Private Event</b>
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
CSTARRouteRequestExtEvent	RouteEvent
CSTAReRouteRequestEvent	ReRouteEvent
CSTARouteUsedExtEvent	RouteUsedEvent
CSTARouteEndEvent	RouteEndEvent
CSTARouteRegisterAbortEvent	RouteCallbackEndedEvent