Preventing Toll Fraud

"Toll fraud" is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or working on your company's behalf). Be aware that there may be a risk of toll fraud associated with your system and that, if toll fraud occurs, it can result in substantial additional charges for your telecommunications services.

Avaya Fraud Intervention

If you suspect that you are being victimized by toll fraud and you need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1 800 643 2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Web site:

http://www.avaya.com

Select Support, then select Escalation Lists US and International. This Web site includes telephone numbers for escalation within the United States. For escalation telephone numbers outside the United States, select Global Escalation List.

Providing Telecommunications Security

Telecommunications security (of voice, data, and/or video communications) is the prevention of any type of intrusion to (that is, either unauthorized or malicious access to or use of) your company's telecommunications equipment by some party. Your company's "telecommunications equipment" includes both this Avaya product and any other voice/data/video equipment that could be accessed via this Avaya product (that is, "networked equipment"). An "outside party" is anyone who is not a corporate employee, agent, subcontractor, or working on your company's behalf. Whereas, a "malicious party" is anyone (including someone who may be otherwise authorized) who accesses your telecommunications equipment with either malicious or mischievous intent.

Such intrusions may be either to/through synchronous (time-multiplexed and/or circuit-based) or asynchronous (character-, message-, or packet-based) equipment or interfaces for reasons of:

- Utilization (of capabilities special to the accessed equipment)
- Theft (such as, of intellectual property, financial assets, or toll-facility access)
- Eavesdropping (privacy invasions to humans)
- Mischief (troubling, but apparently innocuous, tampering)
- Harm (such as harmful tampering, data loss or alteration, regardless of motive or intent)

Be aware that there may be a risk of unauthorized intrusions associated with your system and/or its networked equipment. Also realize that, if such an intrusion should occur, it could result in a variety of losses to your company (including but not limited to, human/data privacy, intellectual property, material assets, financial resources, labor costs, and/or legal costs).

Your Responsibility for Your Company's Telecommunications Security

The final responsibility for securing both this system and its networked equipment rests with you - an Avaya customer's system administrator, your telecommunications peers, and your managers. Base the fulfillment of your responsibility on acquired knowledge and resources from a variety of sources including but not limited to:

- Installation documents
- System administration documents
- Security documents
- Hardware-/software-based security tools
- Shared information between you and your peers
- Telecommunications security experts

To prevent intrusions to your telecommunications equipment, you and your peers should carefully program and configure:

- your Avaya-provided telecommunications systems and their interfaces
- your Avaya-provided software applications, as well as their underlying hardware/software platforms and interfaces
- any other equipment networked to your Avaya products.
Trademarks

- Adobe, Adobe Acrobat, and the Adobe logo are registered trademarks of Adobe Systems, Inc.
- Avaya and MultiVantage is a trademark of Avaya, Inc.
- DEFINITY, and the Avaya logotype are registered trademarks of Avaya, Inc.
- Sun, Sun Microsystems and the Sun logo are registered trademarks and Java, and all Java-based trademarks, Solaris, and Solaris SPARC are trademarks of Sun Microsystems, Inc. in the USA and other countries.
- UNIX is a registered trademark in the USA and other countries, licensed exclusively through X/Open Company Limited.
- All products and company names are trademarks or registered trademarks of their respective holders.

Avaya Support

Avaya provides a telephone number for you to use to report problems or to ask questions about your contact center. The support telephone number is 1-800-242-2121 in the United States. For additional support telephone numbers, see the Avaya Web site:

http://www.avaya.com

Select Support, then select Escalation Lists US and International. This Web site includes telephone numbers for escalation within the United States. For escalation telephone numbers outside the United States, select Global Escalation List.
About This Document

This document is for Java programmers who are developing CTI based client applications. The information on this document complements the Avaya Computer Telephony JTAPI SDK, which can be installed from the Avaya Computer Telephony (Avaya CT) CD-ROM.

About Product Names

The following table describes product name changes that are introduced with this release of the document. In some cases (such as when release numbers are relevant) former product names are retained. In general, the term Avaya MultiVantage Software is synonymous with DEFINITY. For example, a reference such as "Avaya MultiVantage Software servers only" refers to DEFINITY (and Avaya MultiVantage Software servers).

<table>
<thead>
<tr>
<th>New Product Name</th>
<th>Former Product Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avaya™ Computer Telephony (Avaya CT)</td>
<td>CentreVu Computer Telephony (CVCT)</td>
</tr>
<tr>
<td>Avaya MultiVantage Software</td>
<td>DEFINITY</td>
</tr>
</tbody>
</table>

About the Avaya Implementation of JTAPI

- Avaya currently implements JTAPI 1.2, which has a consistant 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 in JTAPI for Avaya CT** — refer to
  
  Section 1: The Avaya Computer Telephony Implementation of JTAPI for All Switches, and, optionally, Chapter 2: Avaya Computer Telephony Extensions to JTAPI

- **Programming in JTAPI for applications used with the Avaya MultiVantage Software and the G3 PBX Driver (G3PD)**, refer to the following parts of this document
  
  Section 1: The Avaya Computer Telephony Implementation of JTAPI for All Switches
  
  Chapter 3: Avaya Computer Telephony Extensions to Avaya MultiVantage Software
  
  optionally,
  
  Section 4, Using Avaya Computer Telephony Private Data Extensions

- **Programming in JTAPI and TSAPI (Switch-Independent)**
  
  - to produce private data packages for applications used with other vendors' switches and their associated drivers, see
    
    Section 4, Using Avaya Computer Telephony Private Data Extensions
  
  - to use or interpret private data for applications used with other vendors' switches and their associated drivers — see
    
    Section 4, Using Avaya Computer Telephony Private Data Extensions
For More Information

Refer to the following documents for background information.

  
  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 latest HTML files, go directly to the web site, http://java.sun.com/products/jtapi.

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

- **Avaya Computer Telephony, Telephony Services Application Programming Interface (TSAPI) for Avaya MultiVantage Programmer's Reference** (DEFPROG.PDF on the Avaya CT CD ROM).
  
  This document describes how Telephony Services and TSAPI support telephony control capabilities in a generic, switch-independent way (that is, support PBXs from various vendors). The architecture allows the incorporation of vendor-specific switch drivers to deliver Telephony Services across various switch environments.

- **Avaya Computer Telephony, Telephony Services Application Programming Interface (TSAPI) Programmer's Reference** (TSAPI.PDF on the Avaya CT CD-ROM).
Section 1

The Avaya Computer Telephony Implementation of JTAPI for All Switches

This section describes the level of support provided by the Avaya Computer Telephony (Avaya CT) implementation of JTAPI for JTAPI interfaces and associated methods. The information in this section applies to:

- any switch that has a driver for the Avaya CT Server.

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

<table>
<thead>
<tr>
<th>Package</th>
<th>JTAPI Interfaces and Methods</th>
<th>Switches Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Package</td>
<td>Address</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>getName</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>getProvider</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>getTerminals</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>getConnections</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>addObserver</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>getObservers</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>removeObserver</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>addCallObserver</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>getCallObservers</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>removeCallObserver</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>getAddressCapabilities</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>AddressObserver</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>addressChangedEvent</td>
<td>All</td>
</tr>
<tr>
<td>Call Center Package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Call Center Capabilities Package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Call Center Events Package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Call Control Package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Call Control Capabilities Package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Call Control Events Package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core Package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core Capabilities Package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core Events Package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Private Data Package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Private Data Capabilities Package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Private Data Events Package</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: If you are an independent switch vendor who is using the JTAPI private data programming environment to develop a private data package for other vendors' switches, or an application programmer who is using or interpreting private data in a raw form, ignore Section 1 (this section) and refer to Section 4, "Using Avaya Computer Telephony Private Data Extensions."
<table>
<thead>
<tr>
<th>Method</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call</td>
<td>All</td>
</tr>
<tr>
<td>getConnection</td>
<td>All</td>
</tr>
<tr>
<td>getProvider</td>
<td>All</td>
</tr>
<tr>
<td>getState</td>
<td>All</td>
</tr>
<tr>
<td>connect</td>
<td>All</td>
</tr>
<tr>
<td>addObserver</td>
<td>All</td>
</tr>
<tr>
<td>getObservers</td>
<td>All</td>
</tr>
<tr>
<td>removeObserver</td>
<td>All</td>
</tr>
<tr>
<td>getCallCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>CallObserver</td>
<td>All</td>
</tr>
<tr>
<td>callChangedEvent</td>
<td>All</td>
</tr>
<tr>
<td>Connection</td>
<td>All</td>
</tr>
<tr>
<td>getState</td>
<td>All</td>
</tr>
<tr>
<td>getCall</td>
<td>All</td>
</tr>
<tr>
<td>getAddress</td>
<td>All</td>
</tr>
<tr>
<td>getTerminalConnections</td>
<td>All</td>
</tr>
<tr>
<td>disconnect</td>
<td>All</td>
</tr>
<tr>
<td>getConnectionCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>JtapiPeer</td>
<td>All</td>
</tr>
<tr>
<td>getName</td>
<td>All</td>
</tr>
<tr>
<td>getServices</td>
<td>All</td>
</tr>
<tr>
<td>getProvider</td>
<td>All</td>
</tr>
<tr>
<td>Provider</td>
<td>All</td>
</tr>
<tr>
<td>getState</td>
<td>All</td>
</tr>
<tr>
<td>getName</td>
<td>All</td>
</tr>
<tr>
<td>getCalls</td>
<td>All</td>
</tr>
<tr>
<td>Method</td>
<td>Category</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>getAddress</td>
<td>All</td>
</tr>
<tr>
<td>getAddresses</td>
<td>All</td>
</tr>
<tr>
<td>getTerminals</td>
<td>All</td>
</tr>
<tr>
<td>getTerminal</td>
<td>All</td>
</tr>
<tr>
<td>shutdown</td>
<td>All</td>
</tr>
<tr>
<td>createCall</td>
<td>All</td>
</tr>
<tr>
<td>addObserver</td>
<td>All</td>
</tr>
<tr>
<td>getObservers</td>
<td>All</td>
</tr>
<tr>
<td>removeObserver</td>
<td>All</td>
</tr>
<tr>
<td>getProviderCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>getCallCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>getConnectionCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>getAddressCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>getTerminalConnectionCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>getTerminalCapabilities</td>
<td>All</td>
</tr>
<tr>
<td><strong>ProviderObserver</strong></td>
<td>All</td>
</tr>
<tr>
<td>providerChangedEvent</td>
<td>All</td>
</tr>
<tr>
<td><strong>Terminal</strong></td>
<td>All</td>
</tr>
<tr>
<td>getName</td>
<td>All</td>
</tr>
<tr>
<td>getProvider</td>
<td>All</td>
</tr>
<tr>
<td>getAddresses2</td>
<td>All</td>
</tr>
<tr>
<td>getTerminalConnections</td>
<td>All</td>
</tr>
<tr>
<td>addObserver</td>
<td>All</td>
</tr>
<tr>
<td>getObservers</td>
<td>All</td>
</tr>
<tr>
<td>removeObserver</td>
<td>All</td>
</tr>
<tr>
<td>addCallObserver</td>
<td>All</td>
</tr>
<tr>
<td>getCallObservers</td>
<td>All</td>
</tr>
<tr>
<td>removeCallObserver</td>
<td>All</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----</td>
</tr>
<tr>
<td>getTerminalCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>TerminalConnection</td>
<td>All</td>
</tr>
<tr>
<td>getState</td>
<td>All</td>
</tr>
<tr>
<td>getTerminal</td>
<td>All</td>
</tr>
<tr>
<td>getConnection</td>
<td>All</td>
</tr>
<tr>
<td>answer</td>
<td>All</td>
</tr>
<tr>
<td>getTerminalConnectionCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>TerminalObserver</td>
<td>All</td>
</tr>
<tr>
<td>terminalChangedEvent</td>
<td>All</td>
</tr>
</tbody>
</table>

**Implementation Notes**

1. For MultiVantage Software, 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:
   ```java
   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:
   ```text
   <tlink>;login=<loginID>;passwd=<pw>
   ```

5. Avaya Computer Telephony allows for an instance of the Telephony Server (Tserver) without the Telephony Services Security Database. (This applies to all versions of Avaya Computer Telephony, R1.2 and later, as well as CentreVu Computer Telephony R 9.5 V1 through R11.1 V1.) The absence of the Telephony Services Security Database (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 (Avaya MultiVantage Software servers and other vendor's switches), Avaya MultiVantage Software servers only, or none (if the implementation is not supported for any switches).

<table>
<thead>
<tr>
<th>JTAPI Interfaces and Methods</th>
<th>Switches Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACDAddress</td>
<td>All</td>
</tr>
<tr>
<td>getLoggedOnAgents</td>
<td>All</td>
</tr>
<tr>
<td>getNumberQueued</td>
<td>All</td>
</tr>
<tr>
<td>getOldestCallQueued</td>
<td>None</td>
</tr>
<tr>
<td>getRelativeQueueLoad</td>
<td>None</td>
</tr>
<tr>
<td>Function</td>
<td>Availability</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>getQueueWaitTime</td>
<td>None</td>
</tr>
<tr>
<td>getACDManagerAddress</td>
<td>None</td>
</tr>
<tr>
<td>ACDAddressObserver</td>
<td>All</td>
</tr>
<tr>
<td>ACDConnection</td>
<td>MultiVantage only</td>
</tr>
<tr>
<td>getACDManagerConnection</td>
<td>MultiVantage only</td>
</tr>
<tr>
<td>ACDManagerAddress</td>
<td>MultiVantage only</td>
</tr>
<tr>
<td>getACDAaddresses</td>
<td>None</td>
</tr>
<tr>
<td>ACDManagerConnection</td>
<td>MultiVantage only</td>
</tr>
<tr>
<td>getACDCollections</td>
<td>MultiVantage only</td>
</tr>
<tr>
<td>AgentObject</td>
<td>All</td>
</tr>
<tr>
<td>setState</td>
<td>All</td>
</tr>
<tr>
<td>getState</td>
<td>All</td>
</tr>
<tr>
<td>getAgentID 3</td>
<td>All</td>
</tr>
<tr>
<td>getACDAddress</td>
<td>All</td>
</tr>
<tr>
<td>getAgentAddress</td>
<td>All</td>
</tr>
<tr>
<td>getAgentAddress</td>
<td>All</td>
</tr>
<tr>
<td>getAgentTerminal</td>
<td>All</td>
</tr>
<tr>
<td>AgentTerminal</td>
<td>All</td>
</tr>
<tr>
<td>addAgent</td>
<td>All</td>
</tr>
<tr>
<td>removeAgent</td>
<td>All</td>
</tr>
<tr>
<td>getAgents</td>
<td>All</td>
</tr>
<tr>
<td>AgentTerminalObserver 4</td>
<td>All</td>
</tr>
<tr>
<td>CallCenterAddress</td>
<td>All</td>
</tr>
<tr>
<td>addCallObserver</td>
<td>All</td>
</tr>
<tr>
<td>CallCenterCall</td>
<td>All</td>
</tr>
<tr>
<td>connectPredictive 5</td>
<td>All</td>
</tr>
<tr>
<td>setApplicationData</td>
<td>None</td>
</tr>
<tr>
<td>Method</td>
<td>Scope</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>getApplicationData</td>
<td>None</td>
</tr>
<tr>
<td>getTrunks</td>
<td>All</td>
</tr>
<tr>
<td>CallCenterCallObserver</td>
<td>All</td>
</tr>
<tr>
<td>CallCenterProvider</td>
<td>All</td>
</tr>
<tr>
<td>getRouteableAddresses</td>
<td>All</td>
</tr>
<tr>
<td>getACDAAddresses</td>
<td>MultiVantage only</td>
</tr>
<tr>
<td>getACDManagerAddresses</td>
<td>MultiVantage only</td>
</tr>
<tr>
<td>CallCenterTrunk</td>
<td>All</td>
</tr>
<tr>
<td>getName</td>
<td>All</td>
</tr>
<tr>
<td>getState</td>
<td>All</td>
</tr>
<tr>
<td>getType</td>
<td>All</td>
</tr>
<tr>
<td>getCall</td>
<td>All</td>
</tr>
<tr>
<td>RouteAddress</td>
<td>All</td>
</tr>
<tr>
<td>registerRouteCallback</td>
<td>All</td>
</tr>
<tr>
<td>cancelRouteCallback</td>
<td>All</td>
</tr>
<tr>
<td>getRouteCallback</td>
<td>All</td>
</tr>
<tr>
<td>getActiveRouteSessions</td>
<td>All</td>
</tr>
<tr>
<td>RouteCallback</td>
<td>All</td>
</tr>
<tr>
<td>routeEvent</td>
<td>All</td>
</tr>
<tr>
<td>reRouteEvent</td>
<td>All</td>
</tr>
<tr>
<td>routeUsedEvent</td>
<td>All</td>
</tr>
<tr>
<td>routeEndEvent</td>
<td>All</td>
</tr>
<tr>
<td>routeCallbackEndedEvent</td>
<td>All</td>
</tr>
<tr>
<td>RouteSession</td>
<td>All</td>
</tr>
<tr>
<td>getRouteAddress</td>
<td>All</td>
</tr>
<tr>
<td>selectRoute</td>
<td>All</td>
</tr>
<tr>
<td>endRoute</td>
<td>All</td>
</tr>
</tbody>
</table>
The **ACDAddress/getLoggedOnAgents** method is fully supported for Avaya MultiVantage Software. 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/getNumberOfQueued** method is fully supported for Avaya MultiVantage Software servers. 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 Avaya MultiVantage Software.

4. The **AgentTerminalObserver** only supports the AgentTermStoppedEv and AgentTermStoppedOffEv 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 Avaya MultiVantage Software and for other non-Avaya switches; however, the answeringEndpointType parameter is not supported. For Avaya MultiVantage Software, the maxRings and answeringTreatment parameters are supported. For other switches, the maxRings and answeringTreatment parameters are not supported.

   - For the Avaya MultiVantage Software servers, 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 getConnection() method.
     - Another way is to use the UserToUserInfo Avaya MultiVantage Software server-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 Avaya MultiVantage Software servers and non-Avaya switches; however, only one RouteCallback may be registered for an Address at a time.

7. The **RouteSession/selectRoute** method is supported for Avaya MultiVantage Software servers and non-Avaya 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.")

<table>
<thead>
<tr>
<th>JTAPI Interfaces and Methods</th>
<th>Supported for All Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACDAddressCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>canGetLoggedOnAgents</td>
<td>All</td>
</tr>
<tr>
<td>canGetNumberQueued</td>
<td>All</td>
</tr>
<tr>
<td>canGetOldestCallQueued</td>
<td>All</td>
</tr>
<tr>
<td>canGetRelativeQueueLoad</td>
<td>All</td>
</tr>
<tr>
<td>canGetQueueWaitTime</td>
<td>All</td>
</tr>
</tbody>
</table>
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 (Avaya MultiVantage Software servers and other vendor's switches), Avaya MultiVantage Software servers 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.

<table>
<thead>
<tr>
<th>JTAPI Interfaces</th>
<th>Switches Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>canGetACDManagerAddress</td>
<td>All</td>
</tr>
<tr>
<td>ACDConnectionCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>canGetACDManagerConnection</td>
<td>All</td>
</tr>
<tr>
<td>ACDManagerAddressCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>canGetACDAddresses</td>
<td>All</td>
</tr>
<tr>
<td>ACDManagerConnectionCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>canGetACDConnections</td>
<td>All</td>
</tr>
<tr>
<td>AgentTerminalCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>canHandleAgents</td>
<td>All</td>
</tr>
<tr>
<td>CallCenterAddressCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>canAddCallObserver</td>
<td>All</td>
</tr>
<tr>
<td>CallCenterCallCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>canConnectPredictive</td>
<td>All</td>
</tr>
<tr>
<td>canHandleApplicationData</td>
<td>All</td>
</tr>
<tr>
<td>canGetTrunks</td>
<td>All</td>
</tr>
<tr>
<td>CallCenterProviderCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>canGetRouteableAddresses</td>
<td>All</td>
</tr>
<tr>
<td>canGetACDAddresses</td>
<td>All</td>
</tr>
<tr>
<td>canGetACDManagerAddresses</td>
<td>All</td>
</tr>
<tr>
<td>canGetTrunks</td>
<td>All</td>
</tr>
<tr>
<td>RouteAddressCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>canRouteCalls</td>
<td>All</td>
</tr>
<tr>
<td>Event Type</td>
<td>Support</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>ACDAddrBusyEv</td>
<td>MultiVantage only</td>
</tr>
<tr>
<td>ACDAddrLoggedOffEv</td>
<td>All</td>
</tr>
<tr>
<td>ACDAddrLoggedOnEv</td>
<td>All</td>
</tr>
<tr>
<td>ACDAddrNotReadyEv</td>
<td>All</td>
</tr>
<tr>
<td>ACDAddrReadyEv</td>
<td>All</td>
</tr>
<tr>
<td>ACDAddrUnknownEv</td>
<td>All</td>
</tr>
<tr>
<td>ACDAddrWorkNotReadyEv</td>
<td>All</td>
</tr>
<tr>
<td>ACDAddrWorkReadyEv</td>
<td>All</td>
</tr>
<tr>
<td>AgentTermBusyEv</td>
<td>MultiVantage only</td>
</tr>
<tr>
<td>AgentTermLoggedOffEv</td>
<td>All</td>
</tr>
<tr>
<td>AgentTermLoggedOnEv</td>
<td>All</td>
</tr>
<tr>
<td>AgentTermNotReadyEv</td>
<td>All</td>
</tr>
<tr>
<td>AgentTermReadyEv</td>
<td>All</td>
</tr>
<tr>
<td>AgentTermUnknownEv</td>
<td>All</td>
</tr>
<tr>
<td>AgentTermWorkNotReadyEv</td>
<td>All</td>
</tr>
<tr>
<td>AgentTermWorkReadyEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCentCallAppDataEv</td>
<td>None</td>
</tr>
<tr>
<td>RouteCallbackEndedEvent</td>
<td>All</td>
</tr>
<tr>
<td>RouteEndEvent</td>
<td>All</td>
</tr>
<tr>
<td>RouteEvent</td>
<td>All</td>
</tr>
<tr>
<td>RouteSessionEvent</td>
<td>All</td>
</tr>
<tr>
<td>RouteUsedEvent</td>
<td>All</td>
</tr>
</tbody>
</table>

**Implementation Notes**

1. These events are not supported for Avaya MultiVantage Software. 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.
<table>
<thead>
<tr>
<th>JTAPI Interfaces and Methods</th>
<th>Switches Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallControlAddress</td>
<td>All</td>
</tr>
<tr>
<td>setForwarding</td>
<td>All</td>
</tr>
<tr>
<td>getForwarding</td>
<td>All</td>
</tr>
<tr>
<td>cancelForwarding</td>
<td>All</td>
</tr>
<tr>
<td>getDoNotDisturb 1</td>
<td>All</td>
</tr>
<tr>
<td>setDoNotDisturb 2</td>
<td>All</td>
</tr>
<tr>
<td>getDoNotDisturb 1</td>
<td>All</td>
</tr>
<tr>
<td>setDoNotDisturb 2</td>
<td>All</td>
</tr>
<tr>
<td>getMessageWaiting</td>
<td>All</td>
</tr>
<tr>
<td>setMessageWaiting</td>
<td>All</td>
</tr>
<tr>
<td>CallControlAddressObserver</td>
<td>All</td>
</tr>
<tr>
<td>CallControlCall</td>
<td>All</td>
</tr>
<tr>
<td>getCallingAddress</td>
<td>All</td>
</tr>
<tr>
<td>getCallingTerminal</td>
<td>All</td>
</tr>
<tr>
<td>getCalledAddress</td>
<td>All</td>
</tr>
<tr>
<td>getLastRedirectedAddress</td>
<td>All</td>
</tr>
<tr>
<td>addParty</td>
<td>MultiVantage only</td>
</tr>
<tr>
<td>drop</td>
<td>All</td>
</tr>
<tr>
<td>offHook</td>
<td>None</td>
</tr>
<tr>
<td>conference</td>
<td>All</td>
</tr>
<tr>
<td>transfer(Call otherCall)</td>
<td>All</td>
</tr>
<tr>
<td>transfer(String address)</td>
<td>None</td>
</tr>
<tr>
<td>setConferenceController</td>
<td>All</td>
</tr>
<tr>
<td>getConferenceController</td>
<td>All</td>
</tr>
<tr>
<td>setTransferController</td>
<td>All</td>
</tr>
<tr>
<td>getTransferController</td>
<td>All</td>
</tr>
<tr>
<td>setConferenceEnable</td>
<td>All</td>
</tr>
<tr>
<td>Method</td>
<td>Scope</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>getConferenceEnable</td>
<td>All</td>
</tr>
<tr>
<td>setTransferEnable</td>
<td>All</td>
</tr>
<tr>
<td>getTransferEnable</td>
<td>All</td>
</tr>
<tr>
<td>consult (TerminalConnection termconn, String address)</td>
<td>All</td>
</tr>
<tr>
<td>consult (TerminalConnection termconn)</td>
<td>None</td>
</tr>
<tr>
<td>CallControlCallObserver</td>
<td>All</td>
</tr>
<tr>
<td>CallControlConnection</td>
<td>All</td>
</tr>
<tr>
<td>getCallControlState</td>
<td>All</td>
</tr>
<tr>
<td>accept</td>
<td>None</td>
</tr>
<tr>
<td>reject</td>
<td>None</td>
</tr>
<tr>
<td>redirect</td>
<td>All</td>
</tr>
<tr>
<td>addToAddress</td>
<td>None</td>
</tr>
<tr>
<td>park</td>
<td>None</td>
</tr>
<tr>
<td>CallControlTerminal</td>
<td>All</td>
</tr>
<tr>
<td>getDoNotDisturb 2</td>
<td>All</td>
</tr>
<tr>
<td>setDoNotDisturb 2</td>
<td>All</td>
</tr>
<tr>
<td>pickup (Connection pickConnection, Address terminalAddress)</td>
<td>All</td>
</tr>
<tr>
<td>pickup (TerminalConnection pickTermConn, Address terminalAddress)</td>
<td>All</td>
</tr>
<tr>
<td>pickup (Address pickAddress, Address terminalAddress)</td>
<td>All</td>
</tr>
<tr>
<td>pickupFromGroup(String pickupGroup, Address terminalAddress)</td>
<td>None</td>
</tr>
<tr>
<td>pickupFromGroup(Address terminalAddress)</td>
<td>All</td>
</tr>
<tr>
<td>CallControlTerminalConnection</td>
<td>All</td>
</tr>
<tr>
<td>getCallControlState</td>
<td>All</td>
</tr>
<tr>
<td>hold</td>
<td>All</td>
</tr>
<tr>
<td>unhold</td>
<td>All</td>
</tr>
<tr>
<td>join</td>
<td>None</td>
</tr>
</tbody>
</table>
**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 Avaya MultiVantage Software servers, 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.”)
<table>
<thead>
<tr>
<th>JTAPI Interface</th>
<th>SSwitches Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>canSetConferenceEnable</td>
<td>All</td>
</tr>
<tr>
<td>canTransfer</td>
<td>All</td>
</tr>
<tr>
<td>canConference</td>
<td>All</td>
</tr>
<tr>
<td>canAddParty</td>
<td>All</td>
</tr>
<tr>
<td>canConsult</td>
<td>All</td>
</tr>
<tr>
<td>CallControlConnectionCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>canRedirect</td>
<td>All</td>
</tr>
<tr>
<td>canAddToAddress</td>
<td>All</td>
</tr>
<tr>
<td>canAccept</td>
<td>All</td>
</tr>
<tr>
<td>canReject</td>
<td>All</td>
</tr>
<tr>
<td>canPark</td>
<td>All</td>
</tr>
<tr>
<td>CallControlTerminalCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>canGetDoNotDisturb</td>
<td>All</td>
</tr>
<tr>
<td>canSetDoNotDisturb</td>
<td>All</td>
</tr>
<tr>
<td>canPickup</td>
<td>All</td>
</tr>
<tr>
<td>canPickupFromGroup</td>
<td>All</td>
</tr>
<tr>
<td>CallControlTerminalConnectionCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>canHold</td>
<td>All</td>
</tr>
<tr>
<td>canUnhold</td>
<td>All</td>
</tr>
<tr>
<td>canJoin</td>
<td>All</td>
</tr>
<tr>
<td>canLeave</td>
<td>All</td>
</tr>
</tbody>
</table>

Support for JTAPI Call Control Events Package

The following table lists each JTAPI interface from the JTAPI Call Control Events Package, (e.g., CallCtlAddrDoNotDisturbEv, CallCtlAddrForwardEv, CallCtlAddrMessageWaitingEv, and so forth), and whether the implementation is supported for all switches (Avaya MultiVantage Software servers and other vendor's switches), Avaya MultiVantage Software servers 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.
<table>
<thead>
<tr>
<th>CallCtlAddrDoNotDisturbEv</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallCtlAddrForwardEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlAddrMessageWaitingEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlConnAlertingEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlConnDialingEv</td>
<td>None</td>
</tr>
<tr>
<td>CallCtlConnDisconnectedEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlConnEstablishedEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlConnFailedEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlConnInitiatedEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlConnNetworkAlertingEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlConnNetworkReachedEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlConnOfferedEv</td>
<td>None</td>
</tr>
<tr>
<td>CallCtlConnQueuedEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlConnUnknownEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlTermConnBridgedEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlTermConnDroppedEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlTermConnHeldEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlTermConnInUseEv</td>
<td>None</td>
</tr>
<tr>
<td>CallCtlTermConnRingingEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlTermConnTalkingEv</td>
<td>All</td>
</tr>
<tr>
<td>CallCtlTermConnUnknownEv</td>
<td>All</td>
</tr>
</tbody>
</table>

Implementation Notes

1. The CallCtlAddrDoNotDisturbEv 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.")
<table>
<thead>
<tr>
<th>JTAPI Interfaces</th>
<th>Supported for All Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddrObservationEndedEv</td>
<td>All</td>
</tr>
<tr>
<td>CallActiveEv</td>
<td>All</td>
</tr>
<tr>
<td>CallInvalidEv</td>
<td>All</td>
</tr>
<tr>
<td>CallObservationEndedEv</td>
<td>All</td>
</tr>
<tr>
<td>ConnAlertingEv</td>
<td>All</td>
</tr>
<tr>
<td>ConnConnectedEv</td>
<td>All</td>
</tr>
</tbody>
</table>

**Support for JTAPI Events Package**

The following table lists each JTAPI interface from the JTAPI Events Package, (e.g., `AddrObservationEndedEv`, `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.
### 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 (Avaya MultiVantage Software servers and other vendor's switches ), Avaya MultiVantage Software servers only, or none (if the implementation is not supported for any switches).

<table>
<thead>
<tr>
<th>JTAPI Interfaces and Methods</th>
<th>Switches Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>MediaCallObserver</td>
<td>MultiVantage only</td>
</tr>
<tr>
<td>MediaTerminalConnection</td>
<td>MultiVantage only</td>
</tr>
<tr>
<td>getMediaAvailability</td>
<td>None</td>
</tr>
</tbody>
</table>
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.")

<table>
<thead>
<tr>
<th>JTAPI Interfaces and Methods</th>
<th>Switches Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>MediaTerminalConnectionCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>canUseDefaultSpeaker</td>
<td>All</td>
</tr>
<tr>
<td>canUseDefaultMicrophone</td>
<td>All</td>
</tr>
<tr>
<td>canUseRecordURL</td>
<td>All</td>
</tr>
<tr>
<td>canUsePlayURL</td>
<td>All</td>
</tr>
<tr>
<td>canStartPlaying</td>
<td>All</td>
</tr>
<tr>
<td>canStopPlaying</td>
<td>All</td>
</tr>
<tr>
<td>canStartRecording</td>
<td>All</td>
</tr>
<tr>
<td>canStopRecording</td>
<td>All</td>
</tr>
<tr>
<td>canDetectDtmf</td>
<td>All</td>
</tr>
<tr>
<td>canGenerateDtmf</td>
<td>All</td>
</tr>
</tbody>
</table>

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 (Avaya MultiVantage Software servers and other vendor's switches), Avaya MultiVantage Software servers 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.

<table>
<thead>
<tr>
<th>JTAPI Interfaces</th>
<th>Supported for All Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>MediaTermConnAvailableEv</td>
<td>None</td>
</tr>
<tr>
<td>MediaTermConnDtmfEv†</td>
<td>MultiVantage only</td>
</tr>
<tr>
<td>MediaTermConnStateEv</td>
<td>None</td>
</tr>
<tr>
<td>MediaTermConnUnavailableEv</td>
<td>None</td>
</tr>
</tbody>
</table>

**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 Avaya MultiVantage Software server-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., getPrivateKeyData, 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.")

<table>
<thead>
<tr>
<th>JTAPI Interfaces and Methods</th>
<th>Switches Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrivateData</td>
<td>All</td>
</tr>
<tr>
<td>getPrivateKeyData</td>
<td>All</td>
</tr>
<tr>
<td>setPrivateKeyData 1</td>
<td>All</td>
</tr>
</tbody>
</table>
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.")

<table>
<thead>
<tr>
<th>JTAPI Interfaces and Methods</th>
<th>Supported for All Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrivateDataCapabilities</td>
<td>All</td>
</tr>
<tr>
<td>canSetPrivateData</td>
<td>All</td>
</tr>
<tr>
<td>canGetPrivateData</td>
<td>All</td>
</tr>
<tr>
<td>canSendPrivateData</td>
<td>All</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>JTAPI Interfaces</th>
<th>Switches Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrivateAddrEv</td>
<td>All</td>
</tr>
<tr>
<td>PrivateCallEv</td>
<td>All</td>
</tr>
<tr>
<td>PrivateProvEv</td>
<td>All</td>
</tr>
<tr>
<td>PrivateTermEv</td>
<td>All</td>
</tr>
</tbody>
</table>
Avaya Computer Telephony Extensions to JTAPI for All switches

The information in this section applies to non-standard additions to JTAPI. This package is available only from the Avaya CT implementation of JTAPI and is not available from any other implementation of JTAPI.

This information is optional. It contains the Avaya Computer Telephony extensions to JTAPI that can be used to program applications for any switch that conforms to the CSTA specification and works in an Avaya CT Server environment.

Who Should Be Using These Extensions?

These extensions are relevant to 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 that has a driver for the Avaya CT Server.

If you are an application programmer who is using JTAPI to develop applications for Avaya MultiVantage Software, also refer to Section 1: "The Avaya Computer Telephony Implementation of JTAPI for All Switches." If you want to take advantage of Avaya MultiVantage Software-specific features that are not accessible through standard JTAPI, refer to Section 3, Avaya Computer Telephony Extensions to Avaya MultiVantage Software.

If you are an independent switch vendor who is using the JTAPI private data programming environment to develop a private data package for other vendors' switches, or an application programmer who is using or interpreting private data in a raw form, ignore this section (Section 2) and refer to Section 4 "Using Avaya Computer Telephony Private Data Extensions."

What are the Extensions?

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

Extensions to JTAPI Exceptions

Avaya CT 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 Avaya Computer Telephony Telephony Services Application Programming Interface (TSAPI) Programmer's Reference (TSAPI.PDF on the Avaya CT CD ROM).

Extensions to JTAPI Provider Events

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

<table>
<thead>
<tr>
<th>TSAPI Provider State</th>
<th>JTAPI Provider State</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITsapiProvider.TSAPI.OUT_OF_SERVICE</td>
<td>Provider.OUT_OF_SERVICE</td>
</tr>
<tr>
<td>ITsapiProvider.TSAPI.INITIALIZING</td>
<td>Provider.OUT_OF_SERVICE</td>
</tr>
<tr>
<td>ITsapiProvider.TSAPI.IN_SERVICE</td>
<td>Provider.IN_SERVICE</td>
</tr>
<tr>
<td>ITsapiProvider.TSAPI.SHUTDOWN</td>
<td>Provider.SHUTDOWN</td>
</tr>
</tbody>
</table>
package com.avaya.jtapi.tsapi

Interface Index

- ITsapiException

Class Index

- TsapiInvalidArgument Exception
- TsapiInvalidPartyException
- TsapiInvalidStateException
- TsapiMethodNotSupportedException
- TsapiPlatformException
- TsapiPrivilegeViolationException
- TsapiProviderUnavailableException
- TsapiResourceUnavailableException

Exception Index
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).

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.

JTAPI
Failed to meet some Jtapi condition.

NORMAL
Error Type of Normal.

getErrorCode()
Returns the error code.

getAddressType()
Returns the error type.

ACS

public static final int ACS
Error Type of ACS.

public static final int CSTA
    Error Type of CSTA.

Error Code implying confirmation is invalid.

Error Code implying Provider is OUT_OF_SERVICE.

Failure is internal to this Jtapi implementation.

Failed to meet some Jtapi condition.

Error Type of Normal.

Methods

getAddressCode

public abstract int getErrorCode()
    Returns the error code.

getAddressType

public abstract int getErrorType()
    Returns the error type.
Class
com.avaya.jtapi.tsapi.TsipInvalidArgumentException

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.

Methods

- **getErrorCode**
  Returns the error code.

- **getErrorType**
  Returns the error type.

Method Index
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

Object

++++ Throwable

++++ Exception

++++ MethodNotSupportedException

++++ 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 to add acs / csta error codes.

getErrorCode

public int getErrorCode()

Returns the error code.

ggetErrorType

public int getErrorType()

Returns the error type.
Class
com.avaya.jtapi.tsapi.TsapiProviderUnavailableException

Object
   +----Throwable
       |   +----Exception
       |       +----RuntimeException
       |       |   +----ProviderUnavailableException
       |       +----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.
Section 3

Avaya Computer Telephony Extensions to Avaya MultiVantage Software

This section describes non-standard additions to JTAPI. This package is available only from the Avaya Computer Telephony (Avaya CT) implementation of JTAPI and is not available from any other implementation of JTAPI.

Who Should Be Using These Extensions?

This information is optional. It is an intermediate private data package that allows programmers to access private data through Java interfaces rather than through raw private data bytes. It contains the Avaya MultiVantage Software specific feature extensions that support the Telephony Services implementation of JTAPI intended for applications that operate solely with the Avaya MultiVantage Software and the associated Avaya CT driver (the Avaya CT G3PBX Driver).

Note:

If you are an application programmer who is using JTAPI to develop applications for any switch for which there is an Avaya Computer Telephony driver, ignore this Section 3 and refer to Section 1, "The Avaya Computer Telephony Implementation of JTAPI for All Switches."

If you want additional TSAPI-specific information that is not accessible through standard JTAPI, refer to "Section 2, Avaya Computer Telephony Extensions to JTAPI for All Switches."

If you are an independent switch vendor who is using the JTAPI private data programming environment to develop a private data package for other vendors' switches, or an application programmer who is using or interpreting private data in a raw form, ignore this section (Section 3) and refer to Section 4, "Using Avaya Computer Telephony Private Data Extensions."

How Should the Extensions be Used?

The Avaya MultiVantage Software specific extensions to JTAPI allow you to use MultiVantage Software features beyond those provided by the standard Telephony Services implementation of JTAPI.

Avaya MultiVantage Software Features Provided by MultiVantage Software Specific Extensions to JTAPI

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

<table>
<thead>
<tr>
<th>Avaya MultiVantage Feature Name and Description</th>
<th>Class or Interface</th>
<th>Returned/Used by Methods in Class or Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advice of Charge</strong> Reports network charges incurred by outgoing trunk calls (supported by G3V5)</td>
<td>LucentChargeAdviceEvent</td>
<td>LucentV5Provider</td>
</tr>
<tr>
<td><strong>Agent Work Mode</strong> 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 &quot;pending&quot; until all current calls active on the Agent's Agent Terminal are completed.</td>
<td>LucentAgent</td>
<td>LucentAgentStateInfo LucentTerminal LucentV5Terminal LucentV5TerminalEx LucentV5AgentStateInfo LucentV6Agent LucentV6AgentStateInfo</td>
</tr>
<tr>
<td><strong>Call Classifier Information</strong></td>
<td>Provides information on call classifier port usage</td>
<td>CallClassifierInfo</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td><strong>Collect Digits</strong></td>
<td>Allows a route request to wait for a specified number of digits to be collected</td>
<td></td>
</tr>
<tr>
<td><strong>Dial-Ahead Digits</strong></td>
<td>Allows a route request to place digits in a dial-ahead buffer</td>
<td></td>
</tr>
<tr>
<td><strong>Direct Agent Calls</strong></td>
<td>Allows calls to be made to and from specific logged-in ACD Agents</td>
<td></td>
</tr>
<tr>
<td><strong>Dropping Resources</strong></td>
<td>Allows specific switch resources to be dropped from the call</td>
<td></td>
</tr>
<tr>
<td><strong>Flexible Billing</strong></td>
<td>Allows changing the billing rate for incoming 900-type calls (supported by G3V5)</td>
<td></td>
</tr>
<tr>
<td><strong>Flexible Generation of DTMF Tones</strong></td>
<td>Enables an application to specify tone duration and inter-tone delay duration.</td>
<td></td>
</tr>
<tr>
<td><strong>Integrated Directory Name</strong></td>
<td>Allows the G3 Integrated Directory Database name to be returned</td>
<td></td>
</tr>
<tr>
<td><strong>Look-Ahead Interflow Information</strong></td>
<td>May be used by a routing server application to determine the proper destination of a call</td>
<td>LookaheadInfo</td>
</tr>
<tr>
<td><strong>Lucent Call Information</strong></td>
<td>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. G3V5 adds support for: Universal Call ID, Originator Type, and Flex Billing Flag.</td>
<td>LucentCallInfo LucentV5CallInfo LucentCallInfo LucentV5CallInfo</td>
</tr>
<tr>
<td><strong>Message Waiting Application Information</strong></td>
<td>Indicates which types of applications have enabled message waiting</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>----------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Network Progress Information</strong></td>
<td>Contains supplementary call progress information from the ISDN Progress Indicator Information Element</td>
<td></td>
</tr>
<tr>
<td>G3V5 adds support for: trunk.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Original Call Information</strong></td>
<td>Contains information about the original call in conjunction with the Call.consult() service.</td>
<td></td>
</tr>
<tr>
<td>G3V5 adds support for: Universal Call ID, Originator Type, and Flex Billing Flag.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Priority Calls</strong></td>
<td>Enables priority calling</td>
<td></td>
</tr>
<tr>
<td><strong>Selective Listen</strong></td>
<td>Allows control of listen paths between parties on a conference call (supported by G3V5)</td>
<td></td>
</tr>
<tr>
<td><strong>Single Step Conference</strong></td>
<td>Adds another party to a call (added party does not alert; used mainly for service observing) (supported by G3V5)</td>
<td></td>
</tr>
<tr>
<td><strong>Supervisor Assist Calls</strong></td>
<td>Allows logged-in ACD Agents to place calls to a supervisor’s extension</td>
<td></td>
</tr>
<tr>
<td><strong>Switch Date and Time Information</strong></td>
<td>Returns the current date and time from the switch</td>
<td></td>
</tr>
<tr>
<td><strong>Trunk Group Information</strong></td>
<td>Provides information on trunk group usage</td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Universal Call ID</strong></td>
<td>A call identifier that is globally unique across switches and the public network (supported by G3V5)</td>
<td></td>
</tr>
<tr>
<td><strong>User Entered Code</strong></td>
<td>The code/digits that may have been entered by the caller through the G3 Call Prompting feature of the Collected Digits feature</td>
<td></td>
</tr>
</tbody>
</table>
**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.

<table>
<thead>
<tr>
<th>UserToUserInfo</th>
<th>LucentCall</th>
<th>LucentCallInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LucentConnection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LucentRouteSession</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LucentTerminalConnection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LucentCallInfo</td>
<td></td>
</tr>
</tbody>
</table>
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.
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.jsapi.jsapi.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.
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.
public abstract interface ITsapiTrunk extends 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
public static final int MWI_MCS
 The message waiting indicator has been enabled via Message Center.

public static final int MWI_PROPMGT
 The message waiting indicator has been enabled via Property Management.

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
The LucentAgent interface extends the ITsapiAgent interface.

- **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.

---

**setStateInfo()**
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
Methods

**getStateInfo**

```java
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**

```java
public abstract void setState(int state,
                               int workMode)
```

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.

- `TsapiInvalidStateException`
  - Implementation determined Agent was in an invalid state for this method.
The LucentCall interface extends ITsapiCall with Lucent-specific features. When a Provider is bound to a Lucent switch, this interface may be used to access additional capabilities.

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

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

See Also:

UserToUserInfo

Method Index

- **connect**(LucentTerminal, LucentAddress, String, boolean, UserToUserInfo)
  Similar to the standard connect(), with the addition of Lucent-specific call parameters.

- **connectDirectAgent**(LucentTerminal, LucentAddress, LucentAgent, boolean, UserToUserInfo)
  Places a direct call to a specific logged-in ACD agent.

- **connectPredictive**(LucentTerminal, LucentAddress, String, int, int, int, int, boolean, UserToUserInfo)
  Similar to the standard connectPredictive(), with the addition of Lucent-specific call parameters.

- **connectSupervisorAssist**(LucentAgent, String, UserToUserInfo)
  Places a call from a logged-in ACD agent to a supervisor's extension.

- **consult**(LucentTerminalConnection, String, boolean, UserToUserInfo)
  Similar to the standard consult(), with the addition of Lucent-specific call parameters.

- **consultDirectAgent**(LucentTerminalConnection, LucentAgent, boolean, UserToUserInfo)
  Places a consultation call with a specific logged-in ACD agent.

- **consultSupervisorAssist**(LucentTerminalConnection, ACDAddress, String, UserToUserInfo)
  Places a consultation call from a logged-in ACD agent to a supervisor's extension.

Methods

- **connect**

  ```java
  public abstract javax.telephony.Connection[] connect(LucentTerminal origterm,
  LucentAddress origaddr,
  String dialedDigits,
  String userToUserInfo
  )
  ```
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 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 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
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, TsapiInvalidArgumentOutOfRangeException, 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**

```java
public abstract javax.telephony.Connection[] connectDirectAgent(LucentTerminal origterm,
origaddr,
calledAgent,
priorityCall,
userInfo,
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**
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`
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.

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_ANSWERVOICE_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**

```java
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**

```java
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**

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

Returns lookahead interflow information associated with the call event.

**getOriginalCallInfo**

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

Returns original call information associated with the call event.

**getReason**

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

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

**getTrunk**

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

Returns the trunk from which the call originated.

**getUserEnteredCode**

```java
```

Returns call prompting digits associated with the call event.

**getUserToUserInfo**

```java
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


Get the supplementary call progress information
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.

**DR_CALL_CLASSIFIER**
Drop a call classifier from the call.

**DR_NONE**

**DR_TONE_GENERATOR**
Drop a tone generator from the call.

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

**DR_CALL_CLASSIFIER**

```java
public static final short DR_CALL_CLASSIFIER

Drop a call classifier from the call.
```

**DR_NONE**

```java
public static final short DR_NONE

Drop a tone generator from the call.
```

**DR_TONE_GENERATOR**

```java
public static final short DR_TONE_GENERATOR

Drop a tone generator from the call.
```

**disconnect**

```java
public static final short DR_CALL_CLASSIFIER

Drop a call classifier from the call.
```
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**

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

Returns information on call classifier port usage.

**getSwitchDateAndTime**

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

Returns current date and time from the switch.

**getTrunkGroupInfo**

```java
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**

```java
public abstract void selectRouteDirectAgent(LucentAgent calledAgent,
                                          boolean priorityCall,
                                          UserToUserInfo userInfo)
```

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**

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

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 ACDAddress specified in the request.

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

At least one of the arguments provided is not valid.

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

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

**getDirectoryName**

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

Return Directory name of this Terminal.
```
interface com.avaya.jtapi.tsapi.LucentTerminalConnection

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 DRNONE.
- 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 DEFINTY 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 DEFINTY 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
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.
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.

### Methods

- **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.

---

**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.

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

- TsapiInvalidStateException
- TsapiMethodNotSupportedException
- TsapiPrivilegeViolationException
- TsapiResourceUnavailableException
- TsapiInvalidArgumentException
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

```java
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

```java
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.
public abstract interface LucentV5Terminal
extends LucentTerminal
The LucentV5Terminal interface extends the LucentTerminal interface.

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.

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 ACDAddress specified in the request.

Throws:  TsapiInvalidArgumentException
At least one of the arguments provided is not valid.

Throws:  TsapiInvalidStateException
Implementation determined AgentTerminal was in an invalid state for this method.
Interface
com.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
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.

```java
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.

```java
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.

removeAgent(Agent, int)

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

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

Object
  +----ASN1
  |    +----ASNSequence
  |    +----LucentPrivateData
  +----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.
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.
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

state
State of Agent.

workMode
Work Mode for Agent.
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
public static final short BT_NEW_RATE
    new rate

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

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
public final class LucentChargeError
extends ASNEnumerated
This class defines constants used with the LucentChargeAdviceEvent.getChargeError() method.
See Also:
  LucentChargeAdviceEvent

- **CE_CHARGE_TOO_LARGE**
  Charge provided by the network is too large

- **CE_LESS_FINAL_CHARGE**
  Final charge provided 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

- **CE_CHARGE_TOO_LARGE**
  public static final short CE_CHARGE_TOO_LARGE
  Charge provided by the network is too large

- **CE_LESS_FINAL_CHARGE**
  public static final short CE_LESS_FINAL_CHARGE
  Final charge provided by the network is less than a previous charge

- **CE_NETWORK_BUSY**
  public static final short 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
public final class LucentChargeType extends ASNEnumerated

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

See Also:
LucentChargeAdviceEvent

- **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)
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

---

**Variables**

- **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.

---

**Public 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.
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
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


   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.
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.
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
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

public static final short UE_ANY

public static final short UE_CALL_PROMPTER

public static final short UE_COLLECT

public static final short UE_DATA_BASE_PROVIDED

public static final short UE_ENTERED

public static final short UE_LOGIN_DIGITS
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

ggetType

public short getType()

Returns the type of digits collected
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
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

trunk

Variables

trunk

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

  ```java
  public boolean canSetBillRate()
  ```
  Returns the Flexible Billing flag, which indicates whether the setBillRate() method is valid for this call

- **getCallOriginatorType**

  ```java
  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:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Identified line - no special treatment</td>
</tr>
</tbody>
</table>
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.
Section 4

Using Avaya Computer Telephony Private Data Extensions

This section describes nonstandard additions to JTAPI. This package is available only from the Avaya Computer Telephony implementation of JTAPI and is not available from any other implementation of JTAPI.

Additionally, this chapter describes the extensions that support the Avaya Computer Telephony implementation of JTAPI for the private data mechanism for other vendors' switches and their associated drivers.

Who Should Be Using These Extensions?

These extensions are appropriate for:

- An independent switch vendor who is using the JTAPI private data programming environment to develop a private data package for switches other than Avaya MultiVantage Software servers.
- 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 "Section 3, Avaya Computer Telephony Extensions to Avaya MultiVantage Software." It is assumed that the reader is familiar with the Java programming language, JTAPI, 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 an Avaya CT driver, ignore this section (Section 4) and refer to Section 1, "The Avaya Computer Telephony Implementation of JTAPI for All Switches." If you want additional TSAPI-specific information that is not accessible through standard JTAPI, refer to Section 2, "Avaya Computer Telephony Extensions to JTAPI for All Switches."

If you are an application programmer who is using JTAPI to develop applications for Avaya MultiVantage Software, ignore this section and refer to Section 1, "The Avaya Computer Telephony Implementation of JTAPI for All Switches." If you want to take advantage of Avaya MultiVantage-specific features that are not accessible through standard JTAPI, refer to Section 3, "Avaya Computer Telephony Extensions to Avaya MultiVantage Software."

How Should the Extensions be Used?

The private data extensions to JTAPI can be used by independent switch vendors to create a private data package for other vendors' switches. Also, these private data extensions allow application programmers to interpret or use private data when their applications are supplied with private data in its raw form (i.e., without an intermediate private data package.)

The remainder of this section provides guidelines for using or interpreting private data when it is supplied in its raw form.

Initialization of Private Data

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:

```java
((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

<table>
<thead>
<tr>
<th>TSAPI Construct</th>
<th>Java Class</th>
<th>JTAPI Object</th>
<th>Conversion Method in TsapiProviderPrivate</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExtendedDeviceID_t</td>
<td>ExtendedDeviceID</td>
<td>Address</td>
<td>getAddress()</td>
</tr>
<tr>
<td>ExtendedDeviceID_t</td>
<td>ExtendedDeviceID</td>
<td>Terminal</td>
<td>getTerminal()</td>
</tr>
<tr>
<td>ConnectionID_t</td>
<td>ConnectionID</td>
<td>Connection</td>
<td>getConnection()</td>
</tr>
<tr>
<td>ConnectionID_t</td>
<td>ConnectionID</td>
<td>TerminalConnection</td>
<td>getTerminalConnection()</td>
</tr>
<tr>
<td>callID (field in a ConnectionID_t)</td>
<td>int</td>
<td>Call</td>
<td>getCall()</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>JTAPI Object</th>
<th>TSAPI Construct</th>
<th>Java Class</th>
<th>Conversion Method in ITsapiConnIDPrivate/ITsapiRoutePrivate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>ConnectionID_t</td>
<td>ConnectionID</td>
<td>getTsapiConnID()</td>
</tr>
<tr>
<td>TerminalConnection</td>
<td>ConnectionID_t</td>
<td>ConnectionID</td>
<td>getTsapiConnID()</td>
</tr>
</tbody>
</table>
| 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.
public abstract interface ITsapiConnIDPrivate

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

See Also:
- ConnectionID

getTsapiConnectionID()

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

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.

### Methods

- **addVendor**(String, String)

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

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

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

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

  To set multiple vendors an application must invoke this method multiple times.
Interface com.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.IN_SERVICE 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.IN_SERVICE 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**
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**

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.

**Methods**

- **getTsapiState**

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

  Returns the TSAPI state of the provider.

- **getVendor**

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

  Returns the data vendor name.

- **getVendorVersion**

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

  Returns the negotiated vendor data version.

- **setDebugPrinting**

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

  Enable/disable debug printing in the debug build version.

- **updateAddresses**

  ```java
  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,
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

```java
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

```java
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.

Methods

- getTsapiState

  public abstract int getTsapiState()
  Returns the Tsapi state associated with this event, ITsapiProvider.TSAPI_OUT_OF_SERVICE.
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.

Methods

getTsapiState

public abstract int getTsapiState()

Returns the Tsapi state associated with this event, ITsapiProvider.TSAPI_SHUTDOWN.
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
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
**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
public static final short EXPLICIT_PUBLIC_UNKNOWN

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

public static final short ID_NOTKNOWN

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

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.

public static final short ID_PROVIDED

Device ID Status of ID_PROVIDED indicates the Device ID is valid

public static final short IMPLICIT_PRIVATE

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

public static final short IMPLICIT_PUBLIC

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

public static final short OTHER_PLAN

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

public static final short TRUNK_GROUP_IDENTIFIER

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

public static final short TRUNK_IDENTIFIER

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

### Constructors

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
public class TsapiPeer
extends Object
implements ITsapiPeer
TsapiPeer implements JtapiPeer.

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

*TsapiPeer*

```java
class TsapiPeer {
    public TsapiPeer()
    {
        This constructor is used by JtapiPeerFactory to create an instance of the peer.
    }
}
```

Methods

*addVendor*

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

*getName*

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

*getProvider*

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

*getServices*

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

### 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

public int tsType

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

This section describes the level of support the Avaya Computer Telephony implementation of JTAPI provides for the private data mechanism for other vendors' switches and their associated drivers. It covers the following topics.

- 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

Intended Audience

This appendix is intended for an independent switch vendor who is using the JTAPI private data programming environment to develop a private data package for other vendors' 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 Section 3, Avaya Computer Telephony Extensions for Avaya MultiVantage Software.)

If you are an application programmer who is using JTAPI to develop applications for any switch that uses an Avaya Computer Telephony driver, ignore this appendix and refer to Section 1, "The Avaya Computer Telephony Implementation of JTAPI for All Switches." If you want additional TSAPI-specific information that is not accessible through standard JTAPI, refer to Section 2, "Avaya Computer Telephony Extensions to JTAPI for All Switches."

If you are an application programmer who is using JTAPI to develop applications for Avaya MultiVantage Software, ignore this appendix and refer to Section 1, "The Avaya Computer Telephony Implementation of JTAPI for All Switches."

If you want to take advantage of Avaya MultiVantage Software specific features that are not accessible through standard JTAPI, refer to Section 3, "Avaya Computer Telephony Extensions to Avaya MultiVantage Software."

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.privatedata.events package. For example, if the desired effect is to get private data that is associated with cstaDeliveredEvent, the way to achieve that in JTAPI is to extract it from the PrivateEvent that is delivered in an event array to a CallObserver.

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

### Mapping of Possible JTAPI Events to TSAPI Events

<table>
<thead>
<tr>
<th>TSAPI Requests</th>
<th>JTAPI Interfaces</th>
<th>JTAPI Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>cstaMakeCall</td>
<td>Call</td>
<td>connect</td>
</tr>
<tr>
<td>cstaClearConnection</td>
<td>Connection</td>
<td>disconnect</td>
</tr>
<tr>
<td>acsEnumServerNames</td>
<td>JtapiPeer</td>
<td>getServices</td>
</tr>
<tr>
<td>acsOpenStream</td>
<td>JtapiPeer</td>
<td>getProvider</td>
</tr>
<tr>
<td>acsCloseStream</td>
<td>Provider</td>
<td>shutdown</td>
</tr>
<tr>
<td>cstaAnswerCall</td>
<td>TerminalConnection</td>
<td>answer</td>
</tr>
<tr>
<td>cstaSetAgentState</td>
<td>AgentTerminal</td>
<td>addAgent</td>
</tr>
<tr>
<td></td>
<td>Agent</td>
<td>setState</td>
</tr>
<tr>
<td>cstaQueryAgentState</td>
<td>Agent</td>
<td>getState</td>
</tr>
<tr>
<td>cstaMakePredictiveCall</td>
<td>CallCenterCall</td>
<td>connectPredictive</td>
</tr>
<tr>
<td>cstaRouteRegisterReq</td>
<td>RouteAddress</td>
<td>registerRouteCallback</td>
</tr>
<tr>
<td>cstaRouteRegisterCancel</td>
<td>RouteAddress</td>
<td>cancelRouteCallback</td>
</tr>
<tr>
<td>cstaRouteSelectInv</td>
<td>RouteSession</td>
<td>selectRoute</td>
</tr>
<tr>
<td>cstaRouteEndInv</td>
<td>RouteSession</td>
<td>endRoute</td>
</tr>
<tr>
<td>cstaSetForwarding</td>
<td>CallControlAddress</td>
<td>setForwarding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cancelForwarding</td>
</tr>
<tr>
<td>cstaQueryForwarding</td>
<td>CallControlAddress</td>
<td>getForwarding</td>
</tr>
<tr>
<td>cstaQueryDoNotDisturb</td>
<td>CallControlAddress</td>
<td>getDoNotDisturb</td>
</tr>
</tbody>
</table>
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.

**TSAPI Requests**

<table>
<thead>
<tr>
<th>TSAPI Requests</th>
<th>Call Control Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>cstaAlternateCall</td>
<td></td>
</tr>
<tr>
<td>cstaCallCompletion</td>
<td></td>
</tr>
<tr>
<td>cstaReconnectCall</td>
<td></td>
</tr>
<tr>
<td>cstaTransferCall</td>
<td></td>
</tr>
<tr>
<td>cstaConsultationCall</td>
<td></td>
</tr>
<tr>
<td>cstaDeflectCall</td>
<td></td>
</tr>
<tr>
<td>cstaQueryDoNotDisturb</td>
<td></td>
</tr>
<tr>
<td>cstaSetDnd</td>
<td></td>
</tr>
<tr>
<td>cstaPickupCall</td>
<td></td>
</tr>
<tr>
<td>cstaGroupPickupCall</td>
<td></td>
</tr>
<tr>
<td>cstaHoldCall</td>
<td></td>
</tr>
<tr>
<td>cstaRetrieveCall</td>
<td></td>
</tr>
<tr>
<td>cstaSendPrivateEvent</td>
<td></td>
</tr>
</tbody>
</table>

**Call Control Services**

- setDoNotDisturb
- getMessageWaiting
- setMessageWaiting
- drop
- conference
- transfer
- consult
- redirect
- getDoNotDisturb
- pickup
- pickupFromGroup
- hold
- unhold
- sendPrivateData

**Supplementary Services**

- queryLastNumber
- queryDeviceInfo

**Monitor Services**

- changeMonitorFilter
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.

<table>
<thead>
<tr>
<th>TSAPI Events</th>
<th>Possible JTAPI Event in Array with Private Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSTACallClearedEvent</td>
<td>CallInvalidEv</td>
</tr>
<tr>
<td>CSTAMonitorEndedEvent</td>
<td>CallObservationEndedEv</td>
</tr>
<tr>
<td>CSTADeliveredEvent</td>
<td>ConnAlertingEv</td>
</tr>
<tr>
<td>CSTAEstablishedEvent</td>
<td>ConnConnectedEv</td>
</tr>
<tr>
<td>CSTAConnectionClearedEvent</td>
<td>ConnDisconnectedEv</td>
</tr>
<tr>
<td>CSTAFailedEvent</td>
<td>ConnFailedEv</td>
</tr>
<tr>
<td>CSTADoNotDisturbEvent</td>
<td>CallCtlAddrDoNotDisturbEv</td>
</tr>
<tr>
<td>CSTAForwardingEvent</td>
<td>CallCtlAddrForwardEv</td>
</tr>
<tr>
<td>CSTAMessageWaitingEvent</td>
<td>CallCtlAddrMessageWaitingEv</td>
</tr>
<tr>
<td>CSTAServiceInitiatedEvent</td>
<td>CallCtlConnInitiatedEv</td>
</tr>
<tr>
<td>CSTANetworkReachedEvent</td>
<td>CallCtlConnNetworkReachedEv</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>CSTAQueuedEvent</td>
<td>CallCtlConnQueuedEv</td>
</tr>
<tr>
<td>CSTALoggedOffEvent</td>
<td>ACDAAddrLoggedOffEv</td>
</tr>
<tr>
<td></td>
<td>AgentTermLoggedOffEv</td>
</tr>
<tr>
<td>CSTALoggedOnEvent</td>
<td>ACDAAddrLoggedOnEv</td>
</tr>
<tr>
<td></td>
<td>AgentTermLoggedOnEv</td>
</tr>
<tr>
<td>CSTANotReadyEvent</td>
<td>ACDAAddrNotReadyEv</td>
</tr>
<tr>
<td></td>
<td>AgentTermNotReadyEv</td>
</tr>
<tr>
<td>CSTAReadyEvent</td>
<td>ACDAAddrReadyEv</td>
</tr>
<tr>
<td></td>
<td>AgentTermReadyEv</td>
</tr>
<tr>
<td>CSTAWorkNotReadyEvent</td>
<td>ACDAAddrWorkNotReadyEv</td>
</tr>
<tr>
<td></td>
<td>AgentTermWorkNotReadyEv</td>
</tr>
<tr>
<td>CSTAWorkReadyEvent</td>
<td>ACDAAddrWorkReadyEv</td>
</tr>
<tr>
<td></td>
<td>AgentTermWorkReadyEv</td>
</tr>
<tr>
<td>CSTARouteRequestExtEvent</td>
<td>RouteEvent</td>
</tr>
<tr>
<td>CSTARouteRequestEvent</td>
<td>ReRouteEvent</td>
</tr>
<tr>
<td>CSTARouteUsedExtEvent</td>
<td>RouteUsedEvent</td>
</tr>
<tr>
<td>CSTARouteEndEvent</td>
<td>RouteEndEvent</td>
</tr>
<tr>
<td>CSTARouteRegisterAbortEvent</td>
<td>RouteCallbackEndedEvent</td>
</tr>
</tbody>
</table>