All Classes

Packages
jain.jcc
jain.jcp

All Classes

CallLoadControlEvent
CallLoadControlListener
Event
EventFilter
InvalidArgumentException
InvalidPartyException
InvalidStateException
JccAddress
JccCall
JccCallEvent
JccCallListener
JccConnection
JccConnectionEvent
JccConnectionListener
JccProvider
JcpAddress
JcpCall
JcpCallEvent
JcpCallListener
JcpConnection
JcpConnectionEvent
JcpConnectionListener
JcpPeer
JcpPeerFactory
JcpPeerUnavailableException
JcpProvider
JcpProviderEvent
JcpProviderListener
MethodNotSupportedException
PlatformException
PrivilegeViolationException
ProviderUnavailableException
ResourceUnavailableException

Overview Package Class Use **Tree** **Deprecated** **Index** **Help**

PREV NEXT **FRAMES** **NO FRAMES**

Packages

jain.jcc	
jain.jcp	

Overview Package Class Use **Tree** **Deprecated** **Index** **Help**

PREV NEXT **FRAMES** **NO FRAMES**

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

All Classes

CallLoadControlEvent
CallLoadControlListener
Event
EventFilter
InvalidArgumentException
InvalidPartyException
InvalidStateException
JccAddress
JccCall
JccCallEvent
JccCallListener
JccConnection
JccConnectionEvent
JccConnectionListener
JccProvider
JcpAddress
JcpCall
JcpCallEvent
JcpCallListener
JcpConnection
JcpConnectionEvent
JcpConnectionListener
JcpPeer
JcpPeerFactory
JcpPeerUnavailableException
JcpProvider
JcpProviderEvent
JcpProviderListener
MethodNotSupportedException
PlatformException
PrivilegeViolationException
ProviderUnavailableException
ResourceUnavailableException

jain.jcc

Interface CallLoadControlEvent

public interface **CallLoadControlEvent**

extends Event

This is the base interface for all Load Control related Events. All events which pertain to Load control must extend this interface. Events which extend this interface are reported via the CallLoadControlListener interface.

Field Summary

static int	**PROVIDER_CALL_OVERLOAD_CEASED** indicates that the network has detected that the overload has ceased.
static int	**PROVIDER_CALL_OVERLOAD_ENCOUNTERED** indicates that the network has detected overload.

Fields inherited from interface jain.jcp.Event

CAUSE_CALL_CANCELLED, CAUSE_DEST_NOT_OBTAINABLE, CAUSE_INCOMPATIBLE_DESTINATION, CAUSE_LOCKOUT, CAUSE_NETWORK_CONGESTION, CAUSE_NETWORK_NOT_OBTAINABLE, CAUSE_NEW_CALL, CAUSE_NORMAL, CAUSE_REDIRECTED, CAUSE_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN

Methods inherited from interface jain.jcp.Event

getCause, getID, getSource

Field Detail

PROVIDER_CALL_OVERLOAD_ENCOUNTERED

`public static final int` **`PROVIDER_CALL_OVERLOAD_ENCOUNTERED`**

indicates that the network has detected overload. This constant indicates a specific event passed via a CallLoadControlEvent event and is reported on the CallLoadControlListener interface.

PROVIDER_CALL_OVERLOAD_CEASED

`public static final int` **`PROVIDER_CALL_OVERLOAD_CEASED`**

indicates that the network has detected that the overload has ceased. This constant indicates a specific event passed via a CallLoadControlEvent event and is reported on the CallLoadControlListener interface.

Overview Package Class Use Tree Deprecated Index Help

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Packages

jain.jcc	
jain.jcp	

Overview Package Class Use **Tree** **Deprecated** **Index** **Help**

PREV NEXT **FRAMES** **NO FRAMES**

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Hierarchy For All Packages

Package Hierarchies:

jain.jcc, jain.jcp

Class Hierarchy

- class java.lang.Object
  - class jain.jcp.**JcpPeerFactory**
  - class java.lang.Throwable (implements java.io.Serializable)
    - class java.lang.Exception
      - class jain.jcp.**InvalidArgumentException**
      - class jain.jcp.**InvalidPartyException**
      - class jain.jcp.**InvalidStateException**
      - class jain.jcp.**JcpPeerUnavailableException**
      - class jain.jcp.**MethodNotSupportedException**
      - class jain.jcp.**PrivilegeViolationException**
      - class jain.jcp.**ResourceUnavailableException**
      - class java.lang.RuntimeException
        - class jain.jcp.**PlatformException**
        - class jain.jcp.**ProviderUnavailableException**

Interface Hierarchy

- interface jain.jcp.**Event**
  - interface jain.jcc.**CallLoadControlEvent**
  - interface jain.jcp.**JcpCallEvent**
    - interface jain.jcc.**JccCallEvent**
      - interface jain.jcc.**JccConnectionEvent**(also extends jain.jcp.JcpConnectionEvent)
    - interface jain.jcp.**JcpConnectionEvent**

- interface jain.jcc.**JccConnectionEvent**(also extends jain.jcc.JccCallEvent)
- interface jain.jcp.**JcpProviderEvent**

- interface jain.jcc.**EventFilter**
- interface java.util.EventListener
  - interface jain.jcc.**CallLoadControlListener**
  - interface jain.jcp.**JcpCallListener**
    - interface jain.jcc.**JccCallListener**
      - interface jain.jcc.**JccConnectionListener**(also extends jain.jcp.JcpConnectionListener)
    - interface jain.jcp.**JcpConnectionListener**
      - interface jain.jcc.**JccConnectionListener**(also extends jain.jcc.JccCallListener)
  - interface jain.jcp.**JcpProviderListener**
- interface jain.jcp.**JcpAddress**
  - interface jain.jcc.**JccAddress**
- interface jain.jcp.**JcpCall**
  - interface jain.jcc.**JccCall**
- interface jain.jcp.**JcpConnection**
  - interface jain.jcc.**JccConnection**
- interface jain.jcp.**JcpPeer**
- interface jain.jcp.**JcpProvider**
  - interface jain.jcc.**JccProvider**

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Deprecated API

Overview Package Class Use Tree Deprecated Index Help
PREV NEXT FRAMES NO FRAMES

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

A

C

D

DISCONNECTED - Static variable in interface jain.jcc.JccConnection

Represents the connection DISCONNECTED state.

E

Event - interface jain.jcp.Event.

The Event interface is the parent of all JCC and JTAPI Event interfaces.

EVENT_BLOCK - Static variable in interface jain.jcc.EventFilter

Predicate return constant: Indicates that the specified event is required and is a blocking Event, that is, call processing will be suspended until the continueProcessing() or any other valid method is called.

EVENT_DISCARD - Static variable in interface jain.jcc.EventFilter

Predicate return constant: Indicates that the specified event is not required.

EVENT_NOTIFY - Static variable in interface jain.jcc.EventFilter

Predicate return constant: Indicates that the specified event is required and is a non-blocking Event (notification only), that is, call processing will not be suspended.

EventFilter - interface jain.jcc.EventFilter.

An instance of this EventFilter is supplied to the event source in the addxxxListener() method by the EventListener to indicate what Events are required by the EventListener.

F

FAILED - Static variable in interface jain.jcc.JccConnection

Represents the FAILED state.

G

getAddress() - Method in interface jain.jcp.JcpConnection

Returns the JcpAddress associated with this JcpConnection.

getAddress(String) - Method in interface jain.jcp.JcpProvider

Returns an Address object which corresponds to the (telephone) number string provided.

getCall() - Method in interface jain.jcp.JcpConnection

Retrieves the Jcpcall that is associated with this Jcpconnection.

getCall() - Method in interface jain.jcp.JcpCallEvent

Returns the JcpCall object associated with this event.

returns the JcpProvider associated with this JcpProvider Event.

getProvider() - Method in interface jain.jcp.JcpCall

Retrieves the provider handling this call object.

getProvider() - Method in interface jain.jcp.JcpAddress

Retrieves the Jcpprovider handling this address object.

getProvider(String) - Method in interface jain.jcp.JcpPeer

Returns an instance of a `Provider` object given a string argument which contains the desired service name.

getServices() - Method in interface jain.jcp.JcpPeer

Returns the services that this implementation supports.

getSource() - Method in interface jain.jcp.Event

Returns the event source of the event.

getState() - Method in interface jain.jcp.JcpProvider

Returns the state of the Jcpprovider.

getState() - Method in interface jain.jcp.JcpCall

Retrieves the state of the call.

getState() - Method in class jain.jcp.InvalidStateException

Returns the state of the object.

getState() - Method in interface jain.jcp.JcpConnection

Retrieves the state of the JcpConnection object.

getType() - Method in interface jain.jcc.JccAddress

Returns the type of this Address object.

getType() - Method in class jain.jcp.InvalidPartyException

Returns the type of party.

getType() - Method in class jain.jcp.PrivilegeViolationException

Returns the type of privilege which is not available.

I

IDLE - Static variable in interface jain.jcc.JccConnection

Represents the connection IDLE state.

IDLE - Static variable in interface jain.jcp.JcpCall

JcpCall.IDLE state indicates the Call has zero Connections.

IN_SERVICE - Static variable in interface jain.jcp.JcpProvider

This state indicates that the JcpProvider is currently available for use.

J

M

N

O

P

indicates that the application will no longer receive JcpProvider Events.

PROVIDER_IN_SERVICE - Static variable in interface jain.jcp.JcpProviderEvent

This indicates that the state of the JcpProvider object has changed to JcpProvider.IN_SERVICE.

PROVIDER_OBJECT - Static variable in class jain.jcp.InvalidStateException

The invalid object in question is the Provider

PROVIDER_OUT_OF_SERVICE - Static variable in interface jain.jcp.JcpProviderEvent

This also indicates that the state of the JcpProvider object has changed to JcpProvider.OUT_OF_SERVICE.

PROVIDER_SHUTDOWN - Static variable in interface jain.jcp.JcpProviderEvent

This also indicates that the state of the JcpProvider object has changed to JcpProvider.SHUTDOWN.

providerCallOverloadCeased(CallLoadControlEvent) - Method in interface jain.jcc.CallLoadControlListener

This method indicates that the network has detected that the overload has ceased and has automatically removed load control on calls requested to a particular address range or calls made to a particular destination.

providerCallOverloadEncountered(CallLoadControlEvent) - Method in interface jain.jcc.CallLoadControlListener

This method indicates that the network has detected overload and may have automatically imposed load control on calls requested to a particular address range or calls made to a particular destination.

providerEventTransmissionEnded(JcpProviderEvent) - Method in interface jain.jcp.JcpProviderListener

Indicates that the application will no longer receive JcpProvider events on the instance of the JcpProviderListener.

providerInService(JcpProviderEvent) - Method in interface jain.jcp.JcpProviderListener

Indicates that the state of the JcpProvider has changed to JcpPROVIDER.IN_SERVICE.

providerOutOfService(JcpProviderEvent) - Method in interface jain.jcp.JcpProviderListener

Indicates that the state of the JcpProvider has changed to JcpProvider.OUT_OF_SERVICE.

providerShutdown(JcpProviderEvent) - Method in interface jain.jcp.JcpProviderListener

Indicates that the state of the JcpProvider has changed to JcpProvider.SHUTDOWN.

ProviderUnavailableException - exception jain.jcp.ProviderUnavailableException.

This exception indicates that the Provider is currently not available to the application.

ProviderUnavailableException() - Constructor for class jain.jcp.ProviderUnavailableException

Constructor with no cause and string.

ProviderUnavailableException(int) - Constructor for class jain.jcp.ProviderUnavailableException

Constructor which takes a cause string.

R

S

T

U

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

How This API Document Is Organized

This API (Application Programming Interface) document has pages corresponding to the items in the navigation bar, described as follows.

Overview

The Overview page is the front page of this API document and provides a list of all packages with a summary for each. This page can also contain an overall description of the set of packages.

Package

Each package has a page that contains a list of its classes and interfaces, with a summary for each. This page can contain four categories:

- Interfaces (italic)
- Classes
- Exceptions
- Errors

Class/Interface

Each class, interface, inner class and inner interface has its own separate page. Each of these pages has three sections consisting of a class/interface description, summary tables, and detailed member descriptions:

- Class inheritance diagram
- Direct Subclasses
- All Known Subinterfaces
- All Known Implementing Classes
- Class/interface declaration
- Class/interface description
- Inner Class Summary
- Field Summary
- Constructor Summary
- Method Summary
- Field Detail

- Constructor Detail
- Method Detail

Each summary entry contains the first sentence from the detailed description for that item. The summary entries are alphabetical, while the detailed descriptions are in the order they appear in the source code. This preserves the logical groupings established by the programmer.

Use

Each documented package, class and interface has its own Use page. This page describes what packages, classes, methods, constructors and fields use any part of the given class or package. Given a class or interface A, its Use page includes subclasses of A, fields declared as A, methods that return A, and methods and constructors with parameters of type A. You can access this page by first going to the package, class or interface, then clicking on the "Use" link in the navigation bar.

Tree (Class Hierarchy)

There is a Class Hierarchy page for all packages, plus a hierarchy for each package. Each hierarchy page contains a list of classes and a list of interfaces. The classes are organized by inheritance structure starting with `java.lang.Object`. The interfaces do not inherit from `java.lang.Object`.

- When viewing the Overview page, clicking on "Tree" displays the hierarchy for all packages.
- When viewing a particular package, class or interface page, clicking "Tree" displays the hierarchy for only that package.

Deprecated API

The Deprecated API page lists all of the API that have been deprecated. A deprecated API is not recommended for use, generally due to improvements, and a replacement API is usually given. Deprecated APIs may be removed in future implementations.

Index

The Index contains an alphabetic list of all classes, interfaces, constructors, methods, and fields.

Prev/Next

These links take you to the next or previous class, interface, package, or related page.

Frames/No Frames

These links show and hide the HTML frames. All pages are available with or without frames.

Serialized Form

Each serializable or externalizable class has a description of its serialization fields and methods. This information is of interest to re-implementors, not to developers using the API. While there is no link in the navigation bar, you can get to this information by going to any serialized class and clicking "Serialized Form" in the "See also" section of the class description.

This help file applies to API documentation generated using the standard doclet.

Overview Package Class Use **Tree** **Deprecated** **Index** **Help**

PREV NEXT **FRAMES** **NO FRAMES**

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Package jain.jcc

Interface Summary

CallLoadControlEvent	This is the base interface for all Load Control related Events.
CallLoadControlListener	Interface for notifying load control related changes happening in a JccProvider event.
EventFilter	An instance of this EventFilter is supplied to the event source in the addxxxListener() method by the EventListener to indicate what Events are required by the EventListener.
JccAddress	This interface represents the JccAddress.
JccCall	The JccCall interface extends the JcpCall interface of JCP.
JccCallEvent	This is the base interface for all JccCall-related events.
JccCallListener	This interface reports all changes to the JccCall object.
JccConnection	A JccConnection object represents a link between a network endpoint (address) and a JccCall object.
JccConnectionEvent	This is the base interface for all JccConnection related events.
JccConnectionListener	This interface is an extension of the JccCallListener and the JcpConnectionListener interface and reports state changes both of the JccCall and its JccConnections.
JccProvider	Provider of JAIN Call Control services.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Package jain.jcc

Packages that use jain.jcc

jain.jcc

Classes in jain.jcc used by jain.jcc

CallLoadControlEvent
This is the base interface for all Load Control related Events.

CallLoadControlListener
Interface for notifying load control related changes happening in a JccProvider event.

EventFilter
An instance of this EventFilter is supplied to the event source in the addxxxListener() method by the EventListener to indicate what Events are required by the EventListener.

JccCallEvent
This is the base interface for all JccCall-related events.

JccCallListener
This interface reports all changes to the JccCall object.

JccConnectionEvent
This is the base interface for all JccConnection related events.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Hierarchy For Package jain.jcc

Package Hierarchies:

All Packages

Interface Hierarchy

- interface jain.jcp.**Event**
  - interface jain.jcc.**CallLoadControlEvent**
  - interface jain.jcp.**JcpCallEvent**
    - interface jain.jcc.**JccCallEvent**
      - interface jain.jcc.**JccConnectionEvent**(also extends jain.jcp.JcpConnectionEvent)
    - interface jain.jcp.**JcpConnectionEvent**
      - interface jain.jcc.**JccConnectionEvent**(also extends jain.jcc.JccCallEvent)
- interface jain.jcc.**EventFilter**
- interface java.util.EventListener
  - interface jain.jcc.**CallLoadControlListener**
  - interface jain.jcp.**JcpCallListener**
    - interface jain.jcc.**JccCallListener**
      - interface jain.jcc.**JccConnectionListener**(also extends jain.jcp.JcpConnectionListener)
    - interface jain.jcp.**JcpConnectionListener**
      - interface jain.jcc.**JccConnectionListener**(also extends jain.jcc.JccCallListener)
- interface jain.jcp.**JcpAddress**
  - interface jain.jcc.**JccAddress**
- interface jain.jcp.**JcpCall**
  - interface jain.jcc.**JccCall**
- interface jain.jcp.**JcpConnection**
  - interface jain.jcc.**JccConnection**
- interface jain.jcp.**JcpProvider**

- interface jain.jcc.**JccProvider**

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Hierarchy For Package jain.jcp

Package Hierarchies:

All Packages

Class Hierarchy

- class java.lang.Object
  - class jain.jcp.**JcpPeerFactory**
  - class java.lang.Throwable (implements java.io.Serializable)
    - class java.lang.Exception
      - class jain.jcp.**InvalidArgumentException**
      - class jain.jcp.**InvalidPartyException**
      - class jain.jcp.**InvalidStateException**
      - class jain.jcp.**JcpPeerUnavailableException**
      - class jain.jcp.**MethodNotSupportedException**
      - class jain.jcp.**PrivilegeViolationException**
      - class jain.jcp.**ResourceUnavailableException**
      - class java.lang.RuntimeException
        - class jain.jcp.**PlatformException**
        - class jain.jcp.**ProviderUnavailableException**

Interface Hierarchy

- interface jain.jcp.**Event**
  - interface jain.jcp.**JcpCallEvent**
    - interface jain.jcp.**JcpConnectionEvent**
  - interface jain.jcp.**JcpProviderEvent**
- interface java.util.EventListener
  - interface jain.jcp.**JcpCallListener**
    - interface jain.jcp.**JcpConnectionListener**

- interface jain.jcp.**JcpProviderListener**
- interface jain.jcp.**JcpAddress**
- interface jain.jcp.**JcpCall**
- interface jain.jcp.**JcpConnection**
- interface jain.jcp.**JcpPeer**
- interface jain.jcp.**JcpProvider**

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Package jain.jcp

Interface Summary

Event	The Event interface is the parent of all JCC and JTAPI Event interfaces.
JcpAddress	An `Address` object represents what we commonly think of as a "telephone number".
JcpCall	A JcpCall is a transient association of (zero or more) addresses for the purposes of engaging in a real-time communications interchange.
JcpCallEvent	This is the base interface for all JcpCall-related events.
JcpCallListener	This interface reports all changes to the Call object.
JcpConnection	Introduction
JcpConnectionEvent	This is the base interface for all JcpConnection related events.
JcpConnectionListener	This interface is an extension of the JcpCallListener interface and reports state changes both of the JcpCall and its JcpConnections.
JcpPeer	The JcpPeer interface represents a vendor's particular implementation of the JCP API.
JcpProvider	A `JcpProvider` represents the telephony software-entity that interfaces with a telephony subsystem.
JcpProviderEvent	This is the base interface for all JcpProvider related Events.
JcpProviderListener	Interface for notifying changes happening in a JcpProvider event.

Class Summary

JcpPeerFactory	The `JcpPeerFactory` class is a class by which applications obtain a JcpProvider object.

Exception Summary

InvalidArgumentException	This Exception indicates that an invalid argument is passed into a method.
InvalidPartyException	This exception indicates that a party given as an argument to the method call was invalid.

InvalidStateException	An InvalidStateException indicates that that current state of an object involved in the method invocation does not meet the acceptable pre-conditions for the method.
JcpPeerUnavailableException	This Exception indicates that the JcpPeer is unavailable on the current system.
MethodNotSupportedException	This Exception indicates that the method which was invoked is not supported by the implementation.
PlatformException	A PlatformException indicates an implementation specific exception.
PrivilegeViolationException	This exception indicates that an action pertaining to a certain object failed because the application did not have the proper security permissions to execute that command.
ProviderUnavailableException	This exception indicates that the Provider is currently not available to the application.
ResourceUnavailableException	This exception indicates that a resource inside the system is not available to complete an operation.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Package jain.jcp

Packages that use jain.jcp

jain.jcc	
jain.jcp	

Classes in jain.jcp used by jain.jcc

Event
The Event interface is the parent of all JCC and JTAPI Event interfaces.

InvalidArgumentException
This Exception indicates that an invalid argument is passed into a method.

InvalidPartyException
This exception indicates that a party given as an argument to the method call was invalid.

InvalidStateException
An InvalidStateException indicates that that current state of an object involved in the method invocation does not meet the acceptable pre-conditions for the method.

JcpAddress
An `Address` object represents what we commonly think of as a "telephone number".

JcpCall
A JcpCall is a transient association of (zero or more) addresses for the purposes of engaging in a real-time communications interchange.

JcpCallEvent
This is the base interface for all JcpCall-related events.

JcpCallListener
This interface reports all changes to the Call object.

JcpConnection
Introduction

JcpConnectionEvent
This is the base interface for all JcpConnection related events.

JcpConnectionListener This interface is an extension of the JcpCallListener interface and reports state changes both of the JcpCall and its JcpConnections.
JcpProvider A `JcpProvider` represents the telephony software-entity that interfaces with a telephony subsystem.
JcpProviderListener Interface for notifying changes happening in a JcpProvider event.
MethodNotSupportedException This Exception indicates that the method which was invoked is not supported by the implementation.
PrivilegeViolationException This exception indicates that an action pertaining to a certain object failed because the application did not have the proper security permissions to execute that command.
ResourceUnavailableException This exception indicates that a resource inside the system is not available to complete an operation.

Classes in jain.jcp used by jain.jcp

Event The Event interface is the parent of all JCC and JTAPI Event interfaces.
InvalidArgumentException This Exception indicates that an invalid argument is passed into a method.
InvalidStateException An InvalidStateException indicates that that current state of an object involved in the method invocation does not meet the acceptable pre-conditions for the method.
JcpAddress An `Address` object represents what we commonly think of as a "telephone number".
JcpCall A JcpCall is a transient association of (zero or more) addresses for the purposes of engaging in a real-time communications interchange.
JcpCallEvent This is the base interface for all JcpCall-related events.
JcpCallListener This interface reports all changes to the Call object.
JcpConnection Introduction

JcpConnectionEvent This is the base interface for all JcpConnection related events.
JcpPeer The JcpPeer interface represents a vendor's particular implementation of the JCP API.
JcpPeerUnavailableException This Exception indicates that the JcpPeer is unavailable on the current system.
JcpProvider A `JcpProvider` represents the telephony software-entity that interfaces with a telephony subsystem.
JcpProviderEvent This is the base interface for all JcpProvider related Events.
JcpProviderListener Interface for notifying changes happening in a JcpProvider event.
MethodNotSupportedException This Exception indicates that the method which was invoked is not supported by the implementation.
PrivilegeViolationException This exception indicates that an action pertaining to a certain object failed because the application did not have the proper security permissions to execute that command.
ProviderUnavailableException This exception indicates that the Provider is currently not available to the application.
ResourceUnavailableException This exception indicates that a resource inside the system is not available to complete an operation.

jain.jcp

Interface Event

All Known Subinterfaces:

CallLoadControlEvent, JccCallEvent, JccConnectionEvent, JcpCallEvent, JcpConnectionEvent, JcpProviderEvent

public interface **Event**

The Event interface is the parent of all JCC and JTAPI Event interfaces. Event interfaces within each package are organized in a hierarchical fashion.

Event objects correspond to the object which is undergoing a state change; the specific state change is conveyed to the application in two ways.

First, the implementation reports the event to a particular method in a particular Listener interface to a listening object;generally the method corresponds to a particular state change.

Second, the event that is presented to the method has an identification integer which indicates the specific state change. The Event.getID() method returns this identification number for each event. The actual event identification integer values that may be conveyed by the individual event object are defined in each of the specific event interfaces.

Each event caries a cause or a reason why the event happened. The Event.getCause() method returns this cause value. The different types of cause values are also defined in this interface.

Field Summary

static int	**CAUSE_CALL_CANCELLED** Cause code indicating the user has terminated call without going on-hook.
static int	**CAUSE_DEST_NOT_OBTAINABLE** Cause code indicating the destination is not available.
static int	**CAUSE_INCOMPATIBLE_DESTINATION** Cause code indicating that a call has encountered an incompatible destination.
static int	**CAUSE_LOCKOUT** Cause code indicating that a call has encountered an inter-digit timeout while dialing.

static int	**CAUSE_NETWORK_CONGESTION** Cause code indicating that a call has encountered network congestion.
static int	**CAUSE_NETWORK_NOT_OBTAINABLE** Cause code indicating that a call could not reach a destination network.
static int	**CAUSE_NEW_CALL** Cause code indicating a new call.
static int	**CAUSE_NORMAL** Cause code indicating a normal operation.
static int	**CAUSE_REDIRECTED** Cause code indicating the cause was because of call being redirected.
static int	**CAUSE_RESOURCES_NOT_AVAILABLE** Cause code indicating that resources were not available.
static int	**CAUSE_SNAPSHOT** Cause code indicating that the event is part of a snapshot of the current state of the call.
static int	**CAUSE_UNKNOWN** Cause code indicating the cause was unknown.

Method Summary

int	**getCause**() Returns the cause associated with this event.
int	**getID**() Returns the id of event.
java.lang.Object	**getSource**() Returns the event source of the event.

Field Detail

CAUSE_NORMAL

```
public static final int CAUSE_NORMAL
```

Cause code indicating a normal operation.

CAUSE_UNKNOWN

`public static final int` **`CAUSE_UNKNOWN`**

Cause code indicating the cause was unknown.

CAUSE_CALL_CANCELLED

`public static final int` **`CAUSE_CALL_CANCELLED`**

Cause code indicating the user has terminated call without going on-hook.

CAUSE_DEST_NOT_OBTAINABLE

`public static final int` **`CAUSE_DEST_NOT_OBTAINABLE`**

Cause code indicating the destination is not available.

CAUSE_INCOMPATIBLE_DESTINATION

`public static final int` **`CAUSE_INCOMPATIBLE_DESTINATION`**

Cause code indicating that a call has encountered an incompatible destination.

CAUSE_LOCKOUT

`public static final int` **`CAUSE_LOCKOUT`**

Cause code indicating that a call has encountered an inter-digit timeout while dialing.

CAUSE_NEW_CALL

`public static final int` **`CAUSE_NEW_CALL`**

Cause code indicating a new call.

CAUSE_RESOURCES_NOT_AVAILABLE

`public static final int` **`CAUSE_RESOURCES_NOT_AVAILABLE`**

Cause code indicating that resources were not available.

CAUSE_NETWORK_CONGESTION

`public static final int` **`CAUSE_NETWORK_CONGESTION`**

Cause code indicating that a call has encountered network congestion.

CAUSE_NETWORK_NOT_OBTAINABLE

`public static final int` **`CAUSE_NETWORK_NOT_OBTAINABLE`**

Cause code indicating that a call could not reach a destination network.

CAUSE_SNAPSHOT

`public static final int` **`CAUSE_SNAPSHOT`**

Cause code indicating that the event is part of a snapshot of the current state of the call.

CAUSE_REDIRECTED

`public static final int` **`CAUSE_REDIRECTED`**

Cause code indicating the cause was because of call being redirected.

Method Detail

getCause

`public int` **`getCause`**`()`

Returns the cause associated with this event. Every event has a cause. The various cause values are defined as public static final variables in this interface.

Returns:

the cause of the event.

---

getID

```
public int getID()
```

Returns the id of event. Every event has an id.

Returns:

the id of the event.

---

getSource

```
public java.lang.Object getSource()
```

Returns the event source of the event.

Returns:

The object sending the event.

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Uses of Interface jain.jcp.Event

Packages that use Event

jain.jcc	
jain.jcp	

Uses of Event in jain.jcc

Subinterfaces of Event in jain.jcc	
interface	**CallLoadControlEvent** This is the base interface for all Load Control related Events.
interface	**JccCallEvent** This is the base interface for all JccCall-related events.
interface	**JccConnectionEvent** This is the base interface for all JccConnection related events.

Uses of Event in jain.jcp

Subinterfaces of Event in jain.jcp	
interface	**JcpCallEvent** This is the base interface for all JcpCall-related events.
interface	**JcpConnectionEvent** This is the base interface for all JcpConnection related events.
interface	**JcpProviderEvent** This is the base interface for all JcpProvider related Events.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcc

Interface JccCallEvent

All Known Subinterfaces:

JccConnectionEvent

public interface **JccCallEvent**
extends JcpCallEvent

This is the base interface for all JccCall-related events.

Field Summary

static int	**CALL_SUPERVISE_END** The CALL_SUPERVISE_END event indicates that the supervision of the call has ended.
static int	**CALL_SUPERVISE_START** The CALL_SUPERVISE_START event indicates that the supervision of the call has started.

Fields inherited from interface jain.jcp.JcpCallEvent

CALL_ACTIVE, CALL_CREATED, CALL_EVENT_TRANSMISSION_ENDED, CALL_INVALID

Fields inherited from interface jain.jcp.Event

CAUSE_CALL_CANCELLED, CAUSE_DEST_NOT_OBTAINABLE, CAUSE_INCOMPATIBLE_DESTINATION, CAUSE_LOCKOUT, CAUSE_NETWORK_CONGESTION, CAUSE_NETWORK_NOT_OBTAINABLE, CAUSE_NEW_CALL, CAUSE_NORMAL, CAUSE_REDIRECTED, CAUSE_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN

Methods inherited from interface jain.jcp.JcpCallEvent

getCall

Methods inherited from interface jain.jcp.Event
getCause, getID, getSource

Field Detail

CALL_SUPERVISE_START

public static final int **CALL_SUPERVISE_START**

The CALL_SUPERVISE_START event indicates that the supervision of the call has started. The policy followed is that given in JccCall.superviseCallReq(..).

CALL_SUPERVISE_END

public static final int **CALL_SUPERVISE_END**

The CALL_SUPERVISE_END event indicates that the supervision of the call has ended. The policy followed was that given in JccCall.superviseCallReq(..).

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcc.JccCallEvent

Packages that use JccCallEvent

jain.jcc

Uses of JccCallEvent in jain.jcc

Subinterfaces of JccCallEvent in jain.jcc	
interface	**JccConnectionEvent** This is the base interface for all JccConnection related events.

Methods in jain.jcc with parameters of type JccCallEvent	
void	**JccCallListener.callSuperviseStart**(JccCallEvent callevent) Indicates that the supervision of the call has started.
void	**JccCallListener.callSuperviseEnd**(JccCallEvent callevent) Indicates that the supervision of the call has ended.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcc

Interface JccConnectionEvent

public interface **JccConnectionEvent**
extends JcpConnectionEvent, JccCallEvent

This is the base interface for all JccConnection related events.

Field Summary

static int	**CONNECTION_ADDRESS_ANALYZE** This event indicates that the state of the JccConnection object has changed to JccConnection.ADDRESS_ANALYZE.
static int	**CONNECTION_ADDRESS_COLLECT** This event indicates that the state of the JccConnection object has changed to JccConnection.ADDRESS_COLLECT.
static int	**CONNECTION_AUTHORIZE_CALL_ATTEMPT** This event indicates that the state of the JccConnection object has changed to JccConnection.AUTHORIZE_CALL_ATTEMPT.
static int	**CONNECTION_CALL_DELIVERY** This event indicates that the state of the JccConnection object has changed to JccConnection.CALL_DELIVERY.
static int	**CONNECTION_SUSPENDED** This event indicates that the state of the JccConnection object has changed to JccConnection.SUSPENDED.

Fields inherited from interface jain.jcp.JcpConnectionEvent

CONNECTION_ALERTING, CONNECTION_CONNECTED, CONNECTION_CREATED, CONNECTION_DISCONNECTED, CONNECTION_FAILED, CONNECTION_INPROGRESS, CONNECTION_UNKNOWN

Fields inherited from interface jain.jcc.JccCallEvent

CALL_SUPERVISE_END, CALL_SUPERVISE_START

Fields inherited from interface jain.jcp.JcpCallEvent
CALL_ACTIVE, CALL_CREATED, CALL_EVENT_TRANSMISSION_ENDED, CALL_INVALID

Fields inherited from interface jain.jcp.Event
CAUSE_CALL_CANCELLED, CAUSE_DEST_NOT_OBTAINABLE, CAUSE_INCOMPATIBLE_DESTINATION, CAUSE_LOCKOUT, CAUSE_NETWORK_CONGESTION, CAUSE_NETWORK_NOT_OBTAINABLE, CAUSE_NEW_CALL, CAUSE_NORMAL, CAUSE_REDIRECTED, CAUSE_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN

Fields inherited from interface jain.jcp.JcpCallEvent
CALL_ACTIVE, CALL_CREATED, CALL_EVENT_TRANSMISSION_ENDED, CALL_INVALID

Fields inherited from interface jain.jcp.Event
CAUSE_CALL_CANCELLED, CAUSE_DEST_NOT_OBTAINABLE, CAUSE_INCOMPATIBLE_DESTINATION, CAUSE_LOCKOUT, CAUSE_NETWORK_CONGESTION, CAUSE_NETWORK_NOT_OBTAINABLE, CAUSE_NEW_CALL, CAUSE_NORMAL, CAUSE_REDIRECTED, CAUSE_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN

Methods inherited from interface jain.jcp.JcpConnectionEvent
getConnection

Field Detail

CONNECTION_AUTHORIZE_CALL_ATTEMPT

`public static final int` **`CONNECTION_AUTHORIZE_CALL_ATTEMPT`**

This event indicates that the state of the JccConnection object has changed to JccConnection.AUTHORIZE_CALL_ATTEMPT.

CONNECTION_ADDRESS_COLLECT

`public static final int` **`CONNECTION_ADDRESS_COLLECT`**

This event indicates that the state of the JccConnection object has changed to JccConnection.ADDRESS_COLLECT.

CONNECTION_ADDRESS_ANALYZE

`public static final int` **`CONNECTION_ADDRESS_ANALYZE`**

This event indicates that the state of the JccConnection object has changed to JccConnection.ADDRESS_ANALYZE.

CONNECTION_CALL_DELIVERY

`public static final int` **`CONNECTION_CALL_DELIVERY`**

This event indicates that the state of the JccConnection object has changed to JccConnection.CALL_DELIVERY.

CONNECTION_SUSPENDED

`public static final int` **`CONNECTION_SUSPENDED`**

This event indicates that the state of the JccConnection object has changed to JccConnection.SUSPENDED.

Uses of Interface jain.jcc.JccConnectionEvent

Packages that use JccConnectionEvent

jain.jcc	

Uses of JccConnectionEvent in jain.jcc

Methods in jain.jcc with parameters of type JccConnectionEvent	
void	**JccConnectionListener.connectionAuthorizeCallAttempt**(JccConnectionEvent connectionevent) Indicates that the JccConnection has just been placed in the JccConnection.AUTHORIZE_CALL_ATTEMPT state
void	**JccConnectionListener.connectionAddressCollect**(JccConnectionEvent connectionevent) Indicates that the JccConnection has just been placed in the JccConnection.ADDRESS_COLLECT state
void	**JccConnectionListener.connectionAddressAnalyze**(JccConnectionEvent connectionevent) Indicates that the JccConnection has just been placed in the JccConnection.ADDRESS_ANALYZE state
void	**JccConnectionListener.connectionCallDelivery**(JccConnectionEvent connectionevent) Indicates that the JccConnection has just been placed in the JccConnection.CALL_DELIVERY state
void	**JccConnectionListener.connectionSuspended**(JccConnectionEvent connectionevent) Indicates that the JccConnection has just been placed in the JccConnection.SUSPENDED state

jain.jcc

Interface JccConnectionListener

public interface **JccConnectionListener**

extends JcpConnectionListener, JccCallListener

This interface is an extension of the JccCallListener and the JcpConnectionListener interface and reports state changes both of the JccCall and its JccConnections.

Method Summary

void	**connectionAddressAnalyze**(JccConnectionEvent connectionevent) Indicates that the JccConnection has just been placed in the JccConnection.ADDRESS_ANALYZE state
void	**connectionAddressCollect**(JccConnectionEvent connectionevent) Indicates that the JccConnection has just been placed in the JccConnection.ADDRESS_COLLECT state
void	**connectionAuthorizeCallAttempt**(JccConnectionEvent connectionevent) Indicates that the JccConnection has just been placed in the JccConnection.AUTHORIZE_CALL_ATTEMPT state
void	**connectionCallDelivery**(JccConnectionEvent connectionevent) Indicates that the JccConnection has just been placed in the JccConnection.CALL_DELIVERY state
void	**connectionSuspended**(JccConnectionEvent connectionevent) Indicates that the JccConnection has just been placed in the JccConnection.SUSPENDED state

Methods inherited from interface jain.jcp.JcpConnectionListener
connectionAlerting, connectionConnected, connectionCreated, connectionDisconnected, connectionFailed, connectionInProgress, connectionUnknown

Methods inherited from interface jain.jcc.JccCallListener
callSuperviseEnd, callSuperviseStart

Method Detail

connectionAuthorizeCallAttempt

`public void` **`connectionAuthorizeCallAttempt`**`(JccConnectionEvent connectionevent)`

Indicates that the JccConnection has just been placed in the JccConnection.AUTHORIZE_CALL_ATTEMPT state

Parameters:

`connectionevent` - JccConnectionEvent.

---

connectionAddressCollect

`public void` **`connectionAddressCollect`**`(JccConnectionEvent connectionevent)`

Indicates that the JccConnection has just been placed in the JccConnection.ADDRESS_COLLECT state

Parameters:

`connectionevent` - JccConnectionEvent.

---

connectionAddressAnalyze

`public void` **`connectionAddressAnalyze`**`(JccConnectionEvent connectionevent)`

Indicates that the JccConnection has just been placed in the JccConnection.ADDRESS_ANALYZE state

Parameters:

`connectionevent` - JccConnectionEvent.

---

connectionCallDelivery

`public void` **`connectionCallDelivery`**`(JccConnectionEvent connectionevent)`

Indicates that the JccConnection has just been placed in the JccConnection.CALL_DELIVERY state

Parameters:

`connectionevent` - JccConnectionEvent.

---

connectionSuspended

`public void` **`connectionSuspended`**`(JccConnectionEvent connectionevent)`

Indicates that the JccConnection has just been placed in the JccConnection.SUSPENDED state

Parameters:

`connectionevent` - JccConnectionEvent.

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcc.JccConnectionListener

No usage of jain.jcc.JccConnectionListener

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcc

Interface JccProvider

public interface **JccProvider**

extends JcpProvider

Provider of JAIN Call Control services. Note also that the JccProvider acts as a Factory to create standard EventFilter objects. These standard EventFilter objects should be provided by the JCC platform implementation. It is hoped that these filters will meet the needs of many applications, thus sparing them of the need to implement them explicitly. It is also possible that by implementing these on the JCC platform (rather than on the application platform) that the cost of remote filter queries can be eliminated thereby addressing the performance problems.

Hence, three standard filters and two filter combiners are proposed. The effect of these three standard filters and two filter combiners is to allow for address ranges in combination with event "masks", an extension of the original event listener proposal. Using these methods, it is possible to create filters that return a given event disposition for address in specific ranges (with holes and overlaps), or for specific events, or a combination of both. It is also possible to make filters that combine standard and custom filters. This would make it possible to quickly determine the filter disposition in many common cases, using standard filters, and only call a custom filter is unusual cases. We later look at each of these standard filters individually.

Fields inherited from interface jain.jcp.JcpProvider
IN_SERVICE, OUT_OF_SERVICE, SHUTDOWN

Method Summary

void	**addCallListener**(JcpCallListener calllistener) Add a call listener to all call objects that will be created under the domain of this provider.
void	**addCallListener**(JcpCallListener calllistener, EventFilter filter) Add a call listener to all call objects that will be created under the domain of this provider.
void	**addCallLoadControlListener**(CallLoadControlListener loadcontrollistener, EventFilter filter) Adds a listener to listen to load control related events.
void	**addConnectionListener**(JcpConnectionListener connectionlistener, EventFilter filter) Add a connection listener to all connections under this JcpProvider.
void	**addProviderListener**(JcpProviderListener providerlistener, EventFilter filter) Adds a listener to this provider.
EventFilter	**createEventFilterAddressRange**(JcpAddress lowAddress, JcpAddress highAddress, int matchDisposition, int nomatchDisposition) This method returns a standard EventFilter which is implemented by the JCC platform.
EventFilter	**createEventFilterAddressRE**(java.lang.String addressRE, int matchDisposition, int nomatchDisposition) This method returns a standard EventFilter which is implemented by the JCC platform.

EventFilter	**createEventFilterAnd**(EventFilter[] filters, int nomatchDisposition) This method returns a standard EventFilter which is implemented by the JCC platform.
EventFilter	**createEventFilterEventSet**(int[] blockEvents, int[] notifyEvents) This method returns a standard EventFilter which is implemented by the JCC platform.
EventFilter	**createEventFilterOr**(EventFilter[] filters, int nomatchDisposition) This method returns a standard EventFilter which is implemented by the JCC platform.
void	**removeCallListener**(JcpCallListener calllistener) Removes a call listener that was registered using JccProvider.addCallListener.
void	**removeCallLoadControlListener**(CallLoadControlListener loadcontrollistener) Deregisters the load control listener.
void	**removeConnectionListener**(JcpConnectionListener connectionlistener) Removes a connection listener that was registered using this.addConnectionListener.
void	**setCallLoadControl**(JcpAddress[] address, double duration, double[] mechanism, int[] treatment) This method imposes or removes load control on calls made to the specified addresses.

Methods inherited from interface jain.jcp.JcpProvider
addProviderListener, createCall, getAddress, getName, getState, removeProviderListener, shutdown

Method Detail

createEventFilterEventSet

```
public EventFilter createEventFilterEventSet(int[] blockEvents,
                                              int[] notifyEvents)
```

This method returns a standard EventFilter which is implemented by the JCC platform. This method takes two arrays of eventID integers (values returned from event.getID()). For event IDs in the blockEvents array, the filter returns EVENT_BLOCK. For event IDs in notifyEvents, the filter returns EVENT_NOTIFY. If any event ID is not listed in one of the three arrays, the filter returns EVENT_DISCARD. The application is supposed to ensure that an event ID is not listed in more than one array. If done, the filter may return any one of the listed event dispositions.

Returns:

EventFilter standard EventFilter provided by the JCC platform to enable filtering of events based on the application's requirements.

---

createEventFilterAddressRange

```
public EventFilter createEventFilterAddressRange(JcpAddress lowAddress,
                                                 JcpAddress highAddress,
                                                 int matchDisposition,
                                                 int nomatchDisposition)
```

This method returns a standard EventFilter which is implemented by the JCC platform. This requires a complete ordering of values in JCPAddress. The ordering is arranged by defining the order to be by JCPAddress.getName()'s string order. For each address in the call obtained by event.getCall(), apply the following. If the address is between lowAddress and

highAddress (inclusive), the filter returns the value matchDisposition. If the address is not in the range specified, then return nomatchDisposition.

Parameters:

`lowAddress` - denotes the JcpAddress which corresponds to the low end of the range.

`highAddress` - denotes the JcpAddress which corresponds to the high end of the range.

`matchDisposition` - indicates the disposition of a JCC related event occurring on a JcpAddress which forms part of the range specified. This should be one of the legal dispositions namely, EVENT_BLOCK, EVENT_DISCARD or EVENT_NOTIFY.

`nomatchDisposition` - indicates the disposition of a JCC related event occurring on a JcpAddress which DOES not form part of the range specified. This should be one of the legal dispositions namely, EVENT_BLOCK, EVENT_DISCARD or EVENT_NOTIFY.

Returns:

EventFilter standard EventFilter provided by the JCC platform to enable filtering of events based on the application's requirements.

---

createEventFilterAddressRE

```
public EventFilter createEventFilterAddressRE(java.lang.String addressRE,
                                              int matchDisposition,
                                              int nomatchDisposition)
```

This method returns a standard EventFilter which is implemented by the JCC platform. This requires a complete ordering of values in JCPAddress. The ordering is arranged by defining the order to be by JCPAddress.getName()'s string order. For each address in the call obtained by event.getCall(), apply the following. Obtain a string using address.getName(). If this string matches the regular expression addressRE, the filter returns the value matchDisposition. If no such addresses are matched, then return nomatchDisposition.

Parameters:

`addressRE` - denotes the regular expression.

`matchDisposition` - indicates the disposition of a JCC related event if the name of the JcpAddress matches the regular expression. This should be one of the legal dispositions namely, EVENT_BLOCK, EVENT_DISCARD or EVENT_NOTIFY.

`nomatchDisposition` - indicates the disposition of a JCC related event if the name of the JcpAddress DOES not matche the regular expression. This should be one of the legal dispositions namely, EVENT_BLOCK, EVENT_DISCARD or EVENT_NOTIFY.

Returns:

EventFilter standard EventFilter provided by the JCC platform to enable filtering of events based on the application's requirements.

---

createEventFilterOr

```
public EventFilter createEventFilterOr(EventFilter[] filters,
                                       int nomatchDisposition)
```

This method returns a standard EventFilter which is implemented by the JCC platform. This filter takes as input an array of EventFilters. For a given event, it applies the filters in order. If a filter returns nomatchDisposition, then the next filter is tested. If a filter returns any other disposition, then the filter returns this value and does no further filter evaluation. This would normally be called with nomatchDisposition set to EVENT_DISCARD to process any event (either by notifying or blocking) that any filterwants to process (logical OR).

Parameters:

`filters` - is an array of EventFilters.

`nomatchDisposition` - indicates the disposition of a JCC related event. This should be one of the legal dispositions namely, EVENT_BLOCK, EVENT_DISCARD or EVENT_NOTIFY.

Returns:

EventFilter standard EventFilter provided by the JCC platform to enable filtering of events based on the application's requirements.

---

createEventFilterAnd

```
public EventFilter createEventFilterAnd(EventFilter[] filters,
                                        int nomatchDisposition)
```

This method returns a standard EventFilter which is implemented by the JCC platform. This filter takes as input an array of EventFilters. For a given event, it applies the filters in order. If the values returned from all filters are the same, then this value is returned as the filter value. Otherwise, the filter returns nomatchDisposition. This means that as soon as any filter returns nomatchDisposition, or as soon as two filters return different values, the filter can immediately return nomatchDisposition. This would normally be called with nomatchDisposition set to EVENT_DISCARD to discard any events that any filter wants to discard (logical AND).

Parameters:

`filters` - is an array of EventFilters.

`nomatchDisposition` - indicates the disposition of a JCC related event. This should be one of the legal dispositions namely, EVENT_BLOCK, EVENT_DISCARD or EVENT_NOTIFY.

Returns:

EventFilter standard EventFilter provided by the JCC platform to enable filtering of events based on the application's requirements.

---

addProviderListener

```
public void addProviderListener(JcpProviderListener providerlistener,
                                EventFilter filter)
                         throws ResourceUnavailableException,
                                MethodNotSupportedException
```

Adds a listener to this provider. Provider related events are reported via the JcpProviderListener interface. The JcpProvider object will report events to this interface for the lifetime of the JcpProvider object or until the listener is removed with `this.removeProviderListener()` method or until the JcpProvider is no longer observable.
Further, this method gives flexibility to the application developers to specify the filtering algorithm explicitly which they would do using the object implementing the filter interface. This way the filtering algorithm can be designed based on the needs of the application.
If the JcpProvider becomes unobservable, a JcpProviderEvent with id PROVIDER_EVENT_TRANSMISSION_ENDED is delivered to the application as a final event. No further events are delivered to the listener unless it is explicitly re-added by the application.

This method is valid anytime and has no pre-conditions. Application must have the ability to add listeners to Providers so they can monitor the changes in state in the Provider. Note that, registering a single listener twice should not result in the listener possibly being notified twice. Instead this will result in replacement of the current filter with the filter specified in the latest addProviderXXX method.

Parameters:

`providerlistener` - JcpProviderListener object that receives the specified events.

`filter` - EventFilter object used to specify the filtering algorithm explicitly and which determines whether the

event is to be sent to the specified listener.

Throws:

MethodNotSupportedException - The listener cannot be added at this time.

ResourceUnavailableException - The resource limit for the number of listeners has been exceeded.

---

addCallListener

```
public void addCallListener(JcpCallListener calllistener)
                    throws MethodNotSupportedException,
                           ResourceUnavailableException
```

Add a call listener to all call objects that will be created under the domain of this provider. This includes changes in the state of the JcpCall and all JcpConnection-related events. The listener added with this method will report events on the call for as long as the implementation can listen to the Call. In the case that the implementation can no longer observe the JcpCall, the applications receives a CALL_EVENT_TRANSMISSION_ENDED. The listener receives no more events after it receives the CALL_EVENT_TRANSMISSION_ENDED.

Listener Lifetime

The `JcpCallListener` will receive events until one of the following occurs, whereupon the listener receives a CALL_EVENT_TRANSMISSION_ENDED.

1. The listener is removed by the application.
2. The implementation can no longer monitor the call.
3. The JcpCall has completed and moved into the `JcpCall.INVALID` state.

Event Snapshots

By default, when an listener is added to a telephone call, the first batch of events may be a "snapshot". That is, if the listener was added after state changes in the Call, the first batch of events will inform the application of the current state of the Call. Note that these snapshot events do NOT provide a history of all events on the Call, rather they provide the minimum necessary information to bring the application up-to-date with the current state of the Call.

Multiple Invocations

If an application attempts to add an instance of an listener already present on this Call, then a repeated invocation will silently fail, i.e. multiple instances of an listener are not added and no exception will be thrown.

Post-Conditions:

1. A snapshot of events is delivered to the listener, if appropriate.

Parameters:

`calllistener` - JcpCallListener object that receives the specified events.

Throws:

MethodNotSupportedException - The listener cannot be added at this time.

ResourceUnavailableException - The resource limit for the number of listeners has been exceeded.

---

addCallListener

```
public void addCallListener(JcpCallListener calllistener,
                            EventFilter filter)
                     throws ResourceUnavailableException,
                            MethodNotSupportedException
```

Add a call listener to all call objects that will be created under the domain of this provider. This includes changes in the state of the JcpCall and all JcpConnection-related events. The listener added with this method will report events on the call for as long as the implementation can listen to the Call. In the case that the implementation can no longer observe the Call, the applications receive a CALL_EVENT_TRANSMISSION_ENDED. The listener receives no more events after it receives the CALL_EVENT_TRANSMISSION_ENDED.

Listener Lifetime

The `CallListener` will receive events until one of the following occurs, whereupon the listener receives a CALL_EVENT_TRANSMISSION_ENDED.

1. The listener is removed by the application.
2. The implementation can no longer monitor the call.
3. The Call has completed and moved into the `JcpCall.INVALID` state.

Event Snapshots

By default, when an listener is added to a telephone call, the first batch of events may be a "snapshot". That is, if the listener was added after state changes in the Call, the first batch of events will inform the application of the current state of the Call. Note that these snapshot events do NOT provide a history of all events on the Call, rather they provide the minimum necessary information to bring the application up-to-date with the current state of the Call.

Multiple Invocations

If an application attempts to add an instance of an listener already present on this Call, then a repeated invocation will silently fail, i.e. multiple instances of an listener are not added and no exception will be thrown. This though results in the filter being added in the last invocation being used by the implementation to filter events.
Post-Conditions:

1. A snapshot of events is delivered to the listener, if appropriate.

Parameters:

`calllistener` - JcpCallListener object that receives the specified events.

`filter` - EventFilter that determines if an event should be delivered to the registered JcpCallListener.

Throws:

MethodNotSupportedException - The listener cannot be added at this time.

ResourceUnavailableException - The resource limit for the number of listeners has been exceeded.

---

removeCallListener

```
public void removeCallListener(JcpCallListener calllistener)
```

Removes a call listener that was registered using JccProvider.addCallListener. The given listener will no longer receive events generated by the Call objects on this Provider. Also, if the listener is not currently registered with the Provider, then this method fails silently, i.e. no listener is removed and no exception is thrown.

Post-Conditions:

1. CALL_EVENT_TRANSMISSION_ENDED is delivered to the application.

Parameters:

`calllistener` - JcpCallListener object to be removed.

---

addConnectionListener

```
public void addConnectionListener(JcpConnectionListener connectionlistener,
                                  EventFilter filter)
```

Add a connection listener to all connections under this JcpProvider.

Note that registering for the same event multiple times should not result in multiple notifications being sent to an application for the same event. Rather, this will result in the last event filter being used to determine if events have to be delivered to the specified ConnectionListener.
Note that this method is also equivalent to this.addCallListener(JcpConnectionListener, filter) since parameter JcpConnectionListener is also a JcpCallListener. However note that JcpCallListeners which are not JcpConnectionListeners cannot be used as a parameter to this method. Thus, this method can be used to add ONLY JcpConnectionListeners and not both JcpConnectionListeners and JcpCallListeners like this.addCallListener().
Post-Conditions:

1. A snapshot of events is delivered to the listener, if appropriate.

Parameters:

`connectionlistener` - JcpConnectionListener object that receives the specified events.
`filter` - EventFilter determines if the ConnectionEvent is to be delivered to the specified listener.

---

removeConnectionListener

```
public void removeConnectionListener(JcpConnectionListener connectionlistener)
```

Removes a connection listener that was registered using this.addConnectionListener. The given listener will no longer receive events generated by the JcpConnection objects related to this JcpProvider object through a JcpCall object. Also, if the listener is not currently registered with the JcpProvider, then this method fails silently, i.e. no listener is removed and no exception is thrown.

Parameters:

`connectionlistener` - JcpConnectionListener object used in the call to addConnectionListener method.

---

setCallLoadControl

```
public void setCallLoadControl(JcpAddress[] address,
                               double duration,
                               double[] mechanism,
                               int[] treatment)
```

```
                    throws MethodNotSupportedException
```

This method imposes or removes load control on calls made to the specified addresses.

The implementation can throw the MethodNotSupportedException if the platform does not support the load control functionality. *Note* that a policy object may be designed to define the policy to be implemented by the platform as a result of this method instead of defining the policy through the given parameters. This might be designed in the future specifications.

Parameters:

`address` - An array of size at most 2. a1[0] denotes the lower address of the range while a1[1] denotes the uper address of the range. Specifying only one element of the array implies that only an individual address is no longer to be the subject of the listener's attention. This constrains the range of addresses added to be numerical addresses. For addresses containing non-numerals such as email addresses, we expect that the application would have to add each address individually. Note that it is expected that adding a range of non-numerical addresses efficiently will be addressed in a future version of this specification.

`duration` - specifies the duration in milliseconds for which the load control should be set. Duration of 0 indicates that the load control should be removed. Duration of -1 indicates an infinite duration (i.e until disabled by the application). Duration of -2 indicates network default duration.

`mechanism` - specifies the load control mechanism to use (such as admitting one call per interval) and any necessary parameters. The contents of this parameter are ignored if the load control duration is set to zero.mech[0] symbolises the call admission rate of the call load control mechanism used. mech[1] symbolises the type of call load control mechanism to use. Thus, mech[0] gives the number of calls to be admitted per interval and mech[1] denotes the interval (in milliseconds) between calls that are admitted.

`treatment` - specifies the treatment of the calls that are not admitted.The contents of this parameter are ignored if the load control duration is set to zero.

Throws:

MethodNotSupportedException - If the implementation does not have load control functionality.

---

addCallLoadControlListener

```
public void addCallLoadControlListener(CallLoadControlListener loadcontrollistener,
                                       EventFilter filter)
                                throws MethodNotSupportedException,
                                       ResourceUnavailableException
```

Adds a listener to listen to load control related events. Note that the load control functionality has to have been specified separately using this.setCallLoadControl(..) method.

Parameters:

`loadcontrollistener` - The listener implementing the CallLoadControlListener interface which will receive all load control related events.

`filter` - EventFilter which specifies if the CallLoadControlEvent is to be delivered to the specified CallLoadControlListener.

Throws:

MethodNotSupportedException - The listener cannot be added at this time.

ResourceUnavailableException - The resource limit for the number of listeners has been exceeded.

---

removeCallLoadControlListener

```
public void
removeCallLoadControlListener(CallLoadControlListener loadcontrollistener)
```

Deregisters the load control listener. This results in the listener not receiving any load control related events in the future. Note that if loadcontrollistener is not already registered using the this.addCallLoadControlListener(..) method then this method fails silently.

Parameters:

`loadcontrollistener` - The listener implementing the CallLoadControlListener interface which will receive all load control related events

---

Overview Package Class Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcc.JccProvider

No usage of jain.jcc.JccProvider

Overview Package Class Use Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

---

jain.jcp

Interface JcpProvider

All Known Subinterfaces:

JccProvider

---

public interface **JcpProvider**

A `JcpProvider` represents the telephony software-entity that interfaces with a telephony subsystem.

Introduction

The telephony subsystem could be a PBX connected to a server machine, a telephony/fax card in a desktop machine or a networking technology such as IP or ATM.

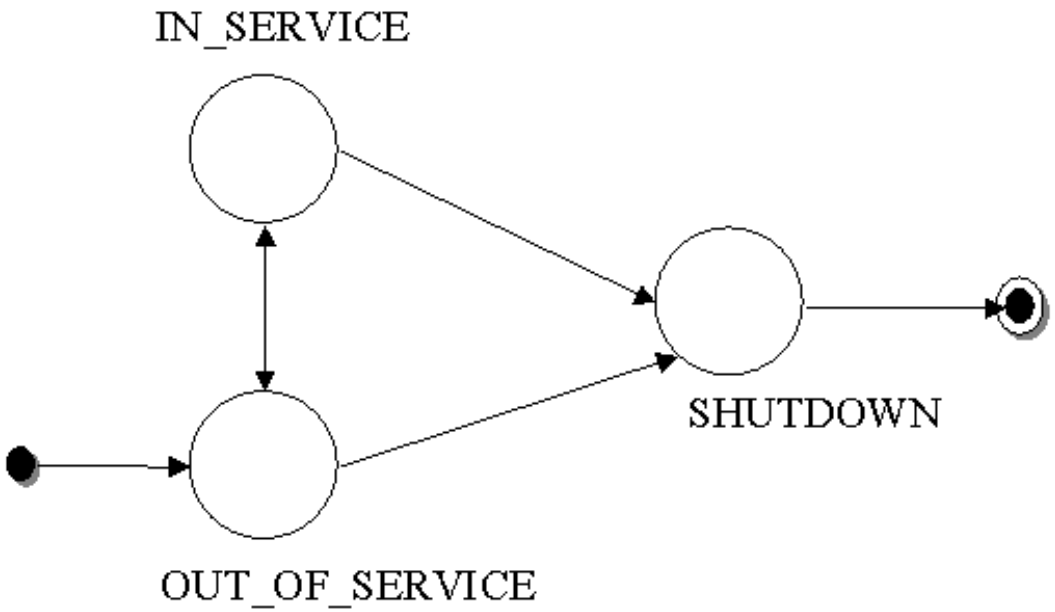
JcpProvider States

The JcpProvider may either be in one of the following states: `JcpProvider.IN_SERVICE`, `JcpProvider.OUT_OF_SERVICE`, or `JcpProvider.SHUTDOWN`. The JcpProvider state represents whether any action on that JcpProvider may be valid. The following tables describes each state:

`JcpProvider.IN_SERVICE`	This state indicates that the JcpProvider is currently alive and available for use.
`JcpProvider.OUT_OF_SERVICE`	This state indicates that a JcpProvider is temporarily not available for use. Many methods in this API are invalid when the JcpProvider is in this state. JcpProviders may come back in service at any time, however, the application can take no direct action to cause this change.
`JcpProvider.SHUTDOWN`:	This state indicates that a JcpProvider is permanently no longer available for use. Most methods in the API are invalid when the JcpProvider is in this state. Applications may use the `JcpProvider.shutdown()` method on this interface to cause a JcpProvider to move into the `JcpProvider.SHUTDOWN` state.

The following diagram shows the allowable state transitions for the JcpProvider.

JCP Provider FSM



Obtaining a JcpProvider

A JcpProvider is created and returned by the `JcpPeer.getProvider()` method which is given a string to describe the desired JcpProvider. This method sets up any needed communication paths between the application and the JcpProvider. The string given is one of the services listed in the `JcpPeer.getServices()`.

Listeners and Events

Each time a state changes occurs on a JcpProvider, the application is notified via an *event*. This event is reported via the `JcpProviderListener` interface. Applications instantiate objects which implement this interface and use the `JcpProvider.addProviderListener()` method to begin the delivery of events. Applications may then query the event object returned for the specific state change, via the `Event.getID()` method. When the JcpProvider changes state, a `JcpProviderEvent` is sent to the JcpProviderListener, having one of the following event ids: `PROVIDER_IN_SERVICE`, `PROVIDER_OUT_OF_SERVICE`, and `PROVIDER_SHUTDOWN`. A `JcpProviderEvent` with event id The `PROVIDER_EVENT_TRANSMISSION_ENDED` is delivered to all JcpProviderListeners when the JcpProvider becomes unobservable and is the final event delivered to the listener.

Call Objects and Providers

Applications may create a JcpCall object representing new calls using the `Provider.createCall()` method. A new JcpCall is returned in the `JcpCall.IDLE` state. Applications may then use this idle JcpCall to place new telephone calls.

Address Objects

A JcpAddress object represents what we commonly think of as a "telephone number." Unlike JcpCall objects, applications may not create JcpAddress objects.

Multiple Providers and Multiple Applications

It is not guaranteed or expected that objects instantiated through one JcpProvider will be usable with another JcpProvider. Therefore, an application that uses two providers must keep all the objects relating to these providers separate. In the future, there may be a mechanism whereby a JcpProvider may share objects with another JcpProvider if they are speaking to the same telephony hardware, however, such capabilities are not available in this release.

Also, multiple applications may request and communicate with the same JcpProvider implementation. Typically, since each application executes in its own object space, each will have its own instance of the JcpProvider object. These two different JcpProvider objects may, in fact, be proxies for a centralized JcpProvider instance. Methods in JCP like `Provider.shutdown()` are specified to affect only the invoking applications and have no affect on others. The only example in the core package is the method.

See Also:

`JcpPeer`, `JcpPeerFactory`, `JcpProviderListener`

Field Summary

static int	**IN_SERVICE** This state indicates that the JcpProvider is currently available for use.
static int	**OUT_OF_SERVICE** This state indicates that the JcpProvider is currently not available for use.
static int	**SHUTDOWN** This state indicates that the JcpProvider is permanently no longer available for use.

Method Summary

void	**addProviderListener**(JcpProviderListener providerlistener) Adds a listener to this provider.
JcpCall	**createCall**() Creates a new instance of the call with no connections.
JcpAddress	**getAddress**(java.lang.String address) Returns an Address object which corresponds to the (telephone) number string provided.
java.lang.String	**getName**() Returns the unique string name of this Provider.
int	**getState**() Returns the state of the Jcpprovider.
void	**removeProviderListener**(JcpProviderListener providerlistener) Removes the given listener from the provider.
void	**shutdown**() Instructs the JcpProvider to shut itself down and provide all necessary cleanup.

Field Detail

IN_SERVICE

```
public static final int IN_SERVICE
```

This state indicates that the JcpProvider is currently available for use.

OUT_OF_SERVICE

```
public static final int OUT_OF_SERVICE
```

This state indicates that the JcpProvider is currently not available for use. Providers may come back in service at any time. However, the application can take no direct action to cause this change.

SHUTDOWN

```
public static final int SHUTDOWN
```

This state indicates that the JcpProvider is permanently no longer available for use.

Method Detail

getState

```
public int getState()
```

Returns the state of the Jcpprovider.

Returns:

Integer representing the state of the provider. See static int's defined in this object.

---

createCall

```
public JcpCall createCall()
                  throws InvalidStateException,
                         ResourceUnavailableException,
                         PrivilegeViolationException,
                         MethodNotSupportedException
```

Creates a new instance of the call with no connections. The new call object is in the JcpCall.IDLE state. An exception is generated if a new call cannot be created for various reasons. This JcpProvider must be in the JcpProvider.IN_SERVICE state, otherwise an InvalidStateException is thrown.

Pre-condition:
1.this.getState() == JcpProvider.IN_SERVICE
Post-conditions:
1.this.getState() == JcpProvider.IN_SERVICE
2.Assume JcpCall call == createCall();
3.call.getState () == JcpCall.IDLE
4.call.getConnections() == null

Returns:

JcpCall object representing the new call.

Throws:

InvalidStateException - If the JcpProvider is not in the JcpProvider.IN_SERVICE state.

ResourceUnavailableException - An internal resource necessary to create a new Call object is unavailable.

PrivilegeViolationException - If the application does not have the proper authority to create a new telephone call object.

MethodNotSupportedException - The implementation does not support creating new JcpCall objects.

---

addProviderListener

```
public void addProviderListener(JcpProviderListener providerlistener)
                         throws ResourceUnavailableException,
                                MethodNotSupportedException
```

Adds a listener to this provider. JcpProvider related events are reported via the JcpProviderListener interface. The JcpProvider object will report events to this interface for the lifetime of the JcpProvider object or until the listener is removed with the JcpProvider.removeProviderListener() method or until the JcpProvider is no longer observable.

If the JcpProvider becomes unobservable, a JcpProviderEvent with id PROVIDER_EVENT_TRANSMISSION_ENDED is delivered to the application as a final event. No further events are delivered to the listener unless it is explicitly re-added by the application.

This method is valid anytime and has no pre-conditions. Application must have the ability to add listeners to JcpProviders so they can monitor the changes in state in the JcpProvider. If an application attempts to add an instance of an listener already present on this JcpProvider, then repeated attempts to add the instance of the listener will silently fail, i.e. multiple instances of an listener are not added and no exception will be thrown.

Parameters:

`providerlistener` - JcpProviderListener object that receives the specified events.

Throws:

MethodNotSupportedException - The listener cannot be added at this time.

ResourceUnavailableException - The resource limit for the number of listeners has been exceeded.

See Also:

`JcpProviderListener`

---

removeProviderListener

```
public void removeProviderListener(JcpProviderListener providerlistener)
```

Removes the given listener from the provider. The given listener will no longer receive events generated by this JcpProvider object. The final event will have id PROVIDER_EVENT_TRANSMISSION_ENDED. Also, if the listener is not currently registered with the JcpProvider, then this method fails silently, i.e. no listener is removed and no exception is thrown.

Post-Conditions:

1. JcpProviderEvent with id PROVIDER_EVENT_TRANSMISSION_ENDED is delivered to listener.

Parameters:

`providerlistener` - JcpProviderListener object being removed.

---

getName

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

Returns the unique string name of this Provider. Each different JcpProvider must have a unique string associated with it. This is the same string which the application passed to the JcpPeer.getProvider() method to create this Provider instance.

Returns:

The unique String name of this Provider.

---

getAddress

`public JcpAddress **getAddress**(java.lang.String address)`
`throws InvalidArgumentException`

Returns an Address object which corresponds to the (telephone) number string provided. If the provided name does not correspond to a JcpAddress known by the JcpProvider and within the JcpProvider's domain, InvalidArgumentException is thrown.

Post-Conditions:

1. Let addres = this.getAddress(addr);
2. Then. (addres.getName()).equals(addr) returns true;

Parameters:

`address` - the address string which possibly represents a telephone number.

Returns:

The JcpAddress object which corresponds to the given number.

Throws:

InvalidArgumentException - If the given number does not correspond to a valid Address under this JcpProvider's domain.

---

shutdown

`public void **shutdown**()`

Instructs the JcpProvider to shut itself down and provide all necessary cleanup. Applications invoke this method when they no longer intend to use the JcpProvider, most often right before they exit. This method is intended to allow the JcpProvider to perform any necessary cleanup which would not be taken care of when the Java objects are garbage collected. This method causes the JcpProvider to move into the JcpProvider.SHUTDOWN state, in which it will stay indefinitely.

If the JcpProvider is already in the JcpProvider.SHUTDOWN state, this method does nothing. The invocation of this method should not affect other applications which are using the same implementation of the JcpProvider object.

Post-Conditions:

1. this.getState ==JcpProvider.SHUTDOWN

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcp.JcpProvider

Packages that use JcpProvider

jain.jcc	
jain.jcp	

Uses of JcpProvider in jain.jcc

Subinterfaces of JcpProvider in jain.jcc	
interface	**JccProvider** Provider of JAIN Call Control services.

Uses of JcpProvider in jain.jcp

Methods in jain.jcp that return JcpProvider	
JcpProvider	**JcpProviderEvent.getProvider()** returns the JcpProvider associated with this JcpProvider Event.
JcpProvider	**JcpCall.getProvider()** Retrieves the provider handling this call object.
JcpProvider	**JcpAddress.getProvider()** Retrieves the Jcpprovider handling this address object.

jain.jcp

Interface JcpProviderEvent

public interface **JcpProviderEvent**

extends Event

This is the base interface for all JcpProvider related Events. All events which pertain to the JcpProvider object must extend this interface. Events which extend this interface are reported via the JcpProviderListener interface.

Field Summary

static int	**PROVIDER_EVENT_TRANSMISSION_ENDED** indicates that the application will no longer receive JcpProvider Events.
static int	**PROVIDER_IN_SERVICE** This indicates that the state of the JcpProvider object has changed to JcpProvider.IN_SERVICE.
static int	**PROVIDER_OUT_OF_SERVICE** This also indicates that the state of the JcpProvider object has changed to JcpProvider.OUT_OF_SERVICE.
static int	**PROVIDER_SHUTDOWN** This also indicates that the state of the JcpProvider object has changed to JcpProvider.SHUTDOWN.

Fields inherited from interface jain.jcp.Event

CAUSE_CALL_CANCELLED, CAUSE_DEST_NOT_OBTAINABLE, CAUSE_INCOMPATIBLE_DESTINATION, CAUSE_LOCKOUT, CAUSE_NETWORK_CONGESTION, CAUSE_NETWORK_NOT_OBTAINABLE, CAUSE_NEW_CALL, CAUSE_NORMAL, CAUSE_REDIRECTED, CAUSE_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN

Method Summary

JcpProvider	**getProvider**() returns the JcpProvider associated with this JcpProvider Event.

Methods inherited from interface jain.jcp.Event
getCause, getID, getSource

Field Detail

PROVIDER_IN_SERVICE

```
public static final int PROVIDER_IN_SERVICE
```

This indicates that the state of the JcpProvider object has changed to JcpProvider.IN_SERVICE.
This constant indicates a specific event passed via a JcpProviderEvent event and is reported on the JcpProviderListener interface.

PROVIDER_OUT_OF_SERVICE

```
public static final int PROVIDER_OUT_OF_SERVICE
```

This also indicates that the state of the JcpProvider object has changed to JcpProvider.OUT_OF_SERVICE.
This constant indicates a specific event passed via a JcpProviderEvent event and is reported on the JcpProviderListener interface.

PROVIDER_SHUTDOWN

```
public static final int PROVIDER_SHUTDOWN
```

This also indicates that the state of the JcpProvider object has changed to JcpProvider.SHUTDOWN.
This constant indicates a specific event passed via a JcpProviderEvent event and is reported on the JcpProviderListener interface.

PROVIDER_EVENT_TRANSMISSION_ENDED

`public static final int` **`PROVIDER_EVENT_TRANSMISSION_ENDED`**

indicates that the application will no longer receive JcpProvider Events.
This constant indicates a specific event passed via a JcpProviderEvent event and is reported on the JcpProviderListener interface.

Method Detail

getProvider

`public` `JcpProvider` **`getProvider`**`()`

returns the JcpProvider associated with this JcpProvider Event.

Returns:

The JcpProvider associated with this event.

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Uses of Interface jain.jcp.JcpProviderEvent

Packages that use JcpProviderEvent

jain.jcp

Uses of JcpProviderEvent in jain.jcp

Methods in jain.jcp with parameters of type JcpProviderEvent	
void	**JcpProviderListener.providerInService**(JcpProviderEvent providerevent) Indicates that the state of the JcpProvider has changed to JcpPROVIDER.IN_SERVICE.
void	**JcpProviderListener.providerOutOfService**(JcpProviderEvent providerevent) Indicates that the state of the JcpProvider has changed to JcpProvider.OUT_OF_SERVICE.
void	**JcpProviderListener.providerShutdown**(JcpProviderEvent providerevent) Indicates that the state of the JcpProvider has changed to JcpProvider.SHUTDOWN.
void	**JcpProviderListener.providerEventTransmissionEnded**(JcpProviderEvent providerevent) Indicates that the application will no longer receive JcpProvider events on the instance of the JcpProviderListener.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Interface JcpProviderListener

public interface **JcpProviderListener**

extends java.util.EventListener

Interface for notifying changes happening in a JcpProvider event. These changes are reported as events to the JcpProviderListener method corresponding to the type of event. Applications must instantiate an object which implements this interface and then use the JcpProvider.addProviderListener() method to register the object to receive all future events associated with the Provider object.

Method Summary

void	**providerEventTransmissionEnded**(JcpProviderEvent providerevent) Indicates that the application will no longer receive JcpProvider events on the instance of the JcpProviderListener.
void	**providerInService**(JcpProviderEvent providerevent) Indicates that the state of the JcpProvider has changed to JcpPROVIDER.IN_SERVICE.
void	**providerOutOfService**(JcpProviderEvent providerevent) Indicates that the state of the JcpProvider has changed to JcpProvider.OUT_OF_SERVICE.
void	**providerShutdown**(JcpProviderEvent providerevent) Indicates that the state of the JcpProvider has changed to JcpProvider.SHUTDOWN.

Method Detail

providerInService

```
public void providerInService(JcpProviderEvent providerevent)
```

Indicates that the state of the JcpProvider has changed to JcpPROVIDER.IN_SERVICE.

Parameters:

providerevent - JcpProviderEvent with event ID JcpProviderEvent.PROVIDER_IN_SERVICE.

providerOutOfService

`public void` **`providerOutOfService`**`(JcpProviderEvent providerevent)`

Indicates that the state of the JcpProvider has changed to JcpProvider.OUT_OF_SERVICE.

Parameters:

`providerevent` - JcpProviderEvent with event ID JcpProviderEvent.PROVIDER_OUT_OF_SERVICE.

---

providerShutdown

`public void` **`providerShutdown`**`(JcpProviderEvent providerevent)`

Indicates that the state of the JcpProvider has changed to JcpProvider.SHUTDOWN.

Parameters:

`providerevent` - JcpProviderEvent with event ID JcpProviderEvent.PROVIDER_SHUTDOWN.

---

providerEventTransmissionEnded

`public void` **`providerEventTransmissionEnded`**`(JcpProviderEvent providerevent)`

Indicates that the application will no longer receive JcpProvider events on the instance of the JcpProviderListener.

Parameters:

`providerevent` - JcpProviderEvent with event ID JcpProviderEvent.PROVIDER_EVENT_TRANSMISSION_ENDED.

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcp.JcpProviderListener

Packages that use JcpProviderListener

jain.jcc	
jain.jcp	

Uses of JcpProviderListener in jain.jcc

Methods in jain.jcc with parameters of type JcpProviderListener	
void	**JccProvider.addProviderListener**(JcpProviderListener providerlistener, EventFilter filter) Adds a listener to this provider.

Uses of JcpProviderListener in jain.jcp

Methods in jain.jcp with parameters of type JcpProviderListener	
void	**JcpProvider.addProviderListener**(JcpProviderListener providerlistener) Adds a listener to this provider.
void	**JcpProvider.removeProviderListener**(JcpProviderListener providerlistener) Removes the given listener from the provider.

jain.jcc

Interface EventFilter

public interface **EventFilter**

An instance of this EventFilter is supplied to the event source in the addxxxListener() method by the EventListener to indicate what Events are required by the EventListener. When an Event occurs, the event source will call the predicate getEventDisposition(Event) to determine if the Event should be fired to the EventListener. Given an event, getEventDisposition() returns

1. **EVENT_DISCARD** if the listener is not interested in receiving the event.
2. **EVENT_NOTIFY** if the listener should be sent a non-blocking notification.
3. **EVENT_BLOCK** if the listener should be sent a blocking event (trigger). This return value applies to JccConnectionEvents only.

The EventFilter while providing flexibility will impact the performance of the platform. Hence, the JCC implementation is expected to provide for some standard EventFilters as explained in the JccProvider interface.

Field Summary

static int	**EVENT_BLOCK** Predicate return constant: Indicates that the specified event is required and is a blocking Event, that is, call processing will be suspended until the continueProcessing() or any other valid method is called.
static int	**EVENT_DISCARD** Predicate return constant: Indicates that the specified event is not required.
static int	**EVENT_NOTIFY** Predicate return constant: Indicates that the specified event is required and is a non-blocking Event (notification only), that is, call processing will not be suspended.

Method Summary

int	**getEventDisposition**(jain.jcc.Event event) This predicate indicates whether the specified Event is required by an EventListener.

Field Detail

EVENT_DISCARD

`public static final int` **`EVENT_DISCARD`**

Predicate return constant: Indicates that the specified event is not required. This is one of the possible return values of getEventDisposition()

EVENT_NOTIFY

`public static final int` **`EVENT_NOTIFY`**

Predicate return constant: Indicates that the specified event is required and is a non-blocking Event (notification only), that is, call processing will not be suspended. This is one of the possible return values of getEventDisposition()

EVENT_BLOCK

`public static final int` **`EVENT_BLOCK`**

Predicate return constant: Indicates that the specified event is required and is a blocking Event, that is, call processing will be suspended until the continueProcessing() or any other valid method is called. This is one of the possible return values of getEventDisposition()

Method Detail

getEventDisposition

`public int` **`getEventDisposition`**`(jain.jcc.Event event)`

This predicate indicates whether the specified Event is required by an EventListener. This method will be called by the Event source prior to firing the event.

Parameters:

`event` - specifies the event.

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcc.EventFilter

Packages that use EventFilter

jain.jcc	

Uses of EventFilter in jain.jcc

Methods in jain.jcc that return EventFilter	
EventFilter	**JccProvider.createEventFilterEventSet**(int[] blockEvents, int[] notifyEvents) This method returns a standard EventFilter which is implemented by the JCC platform.
EventFilter	**JccProvider.createEventFilterAddressRange**(JcpAddress lowAddress, JcpAddress highAddress, int matchDisposition, int nomatchDisposition) This method returns a standard EventFilter which is implemented by the JCC platform.
EventFilter	**JccProvider.createEventFilterAddressRE**(java.lang.String addressRE, int matchDisposition, int nomatchDisposition) This method returns a standard EventFilter which is implemented by the JCC platform.
EventFilter	**JccProvider.createEventFilterOr**(EventFilter[] filters, int nomatchDisposition) This method returns a standard EventFilter which is implemented by the JCC platform.
EventFilter	**JccProvider.createEventFilterAnd**(EventFilter[] filters, int nomatchDisposition) This method returns a standard EventFilter which is implemented by the JCC platform.

Methods in jain.jcc with parameters of type EventFilter	
EventFilter	**JccProvider.createEventFilterOr**(EventFilter[] filters, int nomatchDisposition) This method returns a standard EventFilter which is implemented by the JCC platform.
EventFilter	**JccProvider.createEventFilterAnd**(EventFilter[] filters, int nomatchDisposition) This method returns a standard EventFilter which is implemented by the JCC platform.
void	**JccProvider.addProviderListener**(JcpProviderListener providerlistener, EventFilter filter) Adds a listener to this provider.
void	**JccProvider.addCallListener**(JcpCallListener calllistener, EventFilter filter) Add a call listener to all call objects that will be created under the domain of this provider.
void	**JccProvider.addConnectionListener**(JcpConnectionListener connectionlistener, EventFilter filter) Add a connection listener to all connections under this JcpProvider.
void	**JccProvider.addCallLoadControlListener**(CallLoadControlListener loadcontrollistener, EventFilter filter) Adds a listener to listen to load control related events.
void	**JccCall.addCallListener**(JcpCallListener calllistener, EventFilter filter) Add a listener to this call.

void	**JccCall.addConnectionListener**(JcpConnectionListener cl, EventFilter filter) Add a connection listener to all connections under this call.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Interface JcpAddress

All Known Subinterfaces:

JccAddress

public interface **JcpAddress**

An `Address` object represents what we commonly think of as a "telephone number".

Introduction

An address uniquely identifies a communication endpoint--physical or logical. Its string representation is obtained by the method `JcpAddress.getName()`.

Address objects may be classified into two categories: *local* and *remote*. A "Local" Address is one physically or administratively serviced by the existing Provider. Conversely, a "Remote" address is not served by this Provider. A remote Address may not have the full visibility or control capabilities of a local address. (By impaired visibility, we mean connection states involving this address may not be as finely discriminated; by impaired control, we mean that not all the connection's methods involving this address may be functional.) Note that applications never explicitly create new Address objects.

Address and Call Objects

Address objects are related to Call objects via the Connection object. The Connection object has a *state* which describes the current relationship between the Call and the Address. Each Address object may be part of more than one telephone call, and in each case, is represented by a separate Connection object.

An Address is associated with a Call until the Connection moves into the `DISCONNECTED` state.

Method Summary

`java.lang.String`	**`getName()`** Returns the string representation of the JcpAddress.
`JcpProvider`	**`getProvider()`** Retrieves the Jcpprovider handling this address object.

Method Detail

getName

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

Returns the string representation of the JcpAddress. Note that each JcpAddress possesses a unique string representation within a given JcpProvider.

Returns:

the "unique" string representation of this JcpAddress.

---

getProvider

`public` `JcpProvider` **`getProvider`**`()`

Retrieves the Jcpprovider handling this address object. This JcpProvider object is valid throughout the lifetime of the JcpAddress and does not change once the JcpAddress is created.

Returns:

JcpProvider object managing this call.

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcp.JcpAddress

Packages that use JcpAddress

jain.jcc	
jain.jcp	

Uses of JcpAddress in jain.jcc

Subinterfaces of JcpAddress in jain.jcc	
interface	**JccAddress** This interface represents the JccAddress.

Methods in jain.jcc that return JcpAddress	
JcpAddress	**JccConnection.getLastAddr()** Returns the last redirected JcpAddress associated with this JcpCall.
JcpAddress	**JccConnection.getOriginalAddress()** Returns the original JcpAddress associated with this JcpCall.

Methods in jain.jcc with parameters of type JcpAddress	
EventFilter	**JccProvider.createEventFilterAddressRange**(JcpAddress lowAddress, JcpAddress highAddress, int matchDisposition, int nomatchDisposition) This method returns a standard EventFilter which is implemented by the JCC platform.
void	**JccProvider.setCallLoadControl**(JcpAddress[] address, double duration, double[] mechanism, int[] treatment) This method imposes or removes load control on calls made to the specified addresses.

Uses of JcpAddress in jain.jcp

Methods in jain.jcp that return JcpAddress	
JcpAddress	**JcpProvider.getAddress**(java.lang.String address) Returns an Address object which corresponds to the (telephone) number string provided.
JcpAddress	**JcpConnection.getAddress**() Returns the JcpAddress associated with this JcpConnection.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcc

Interface JccAddress

public interface **JccAddress**
extends JcpAddress

This interface represents the JccAddress.

Method Summary

java.lang.String	**getType**() Returns the type of this Address object.

Methods inherited from interface jain.jcp.JcpAddress
getName, getProvider

Method Detail

getType

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

Returns the type of this Address object. The type of Address can denote whether it is an IP address or a telephone address with a particular numbering scheme.

Returns:

the type of this Address object.

Uses of Interface jain.jcc.JccAddress

No usage of jain.jcc.JccAddress

Overview Package Class Use Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

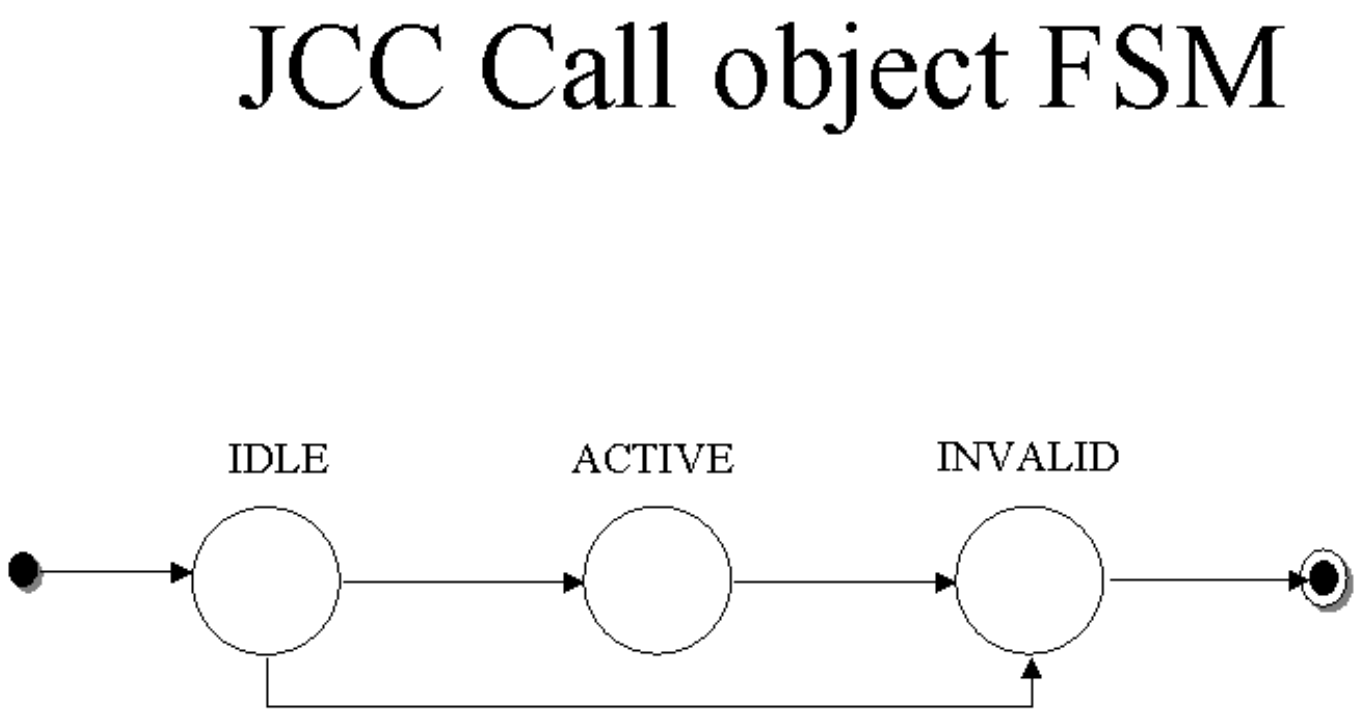
jain.jcc

Interface JccCall

public interface **JccCall**

extends JcpCall

The JccCall interface extends the JcpCall interface of JCP. This interface provides additional methods on a Call. Further, the state machine on the JccCall is also similar to the state machine of the JcpCall except for an extra transition as shown in the following figure.



Fields inherited from interface jain.jcp.JcpCall
ACTIVE, IDLE, INVALID

Method Summary

void	**addCallListener**(JcpCallListener calllistener, EventFilter filter) Add a listener to this call.
void	**addConnectionListener**(JcpConnectionListener cl, EventFilter filter) Add a connection listener to all connections under this call.
JcpConnection	**createConnection**(java.lang.String targetAddress, java.lang.String originatingAddress, java.lang.String originalCalledAddress, java.lang.String redirectingAddress) Creates a new JccConnection and attaches it to this JccCall.
void	**release**() This method requests the release of the call object and associated connection objects.
void	**removeConnectionListener**(JcpConnectionListener cl) Removes the connection listener from all connections under this call.
JcpConnection	**routeCall**(java.lang.String targetAddress, java.lang.String originatingAddress, java.lang.String originalDestinationAddress, java.lang.String redirectingAddress) This method requests routing of a call to the given call party.
void	**superviseCall**(JccCallListener calllistener, double time, int treatment, double bytes) The application calls this method to supervise a call.

Methods inherited from interface jain.jcp.JcpCall
addCallListener, getConnections, getProvider, getState, removeCallListener

Method Detail

addCallListener

```
public void addCallListener(JcpCallListener calllistener,
                            EventFilter filter)
                     throws ResourceUnavailableException,
                            MethodNotSupportedException
```

Add a listener to this call. This also reports all state changes in the state of the JccCall and JccConnection objects. The listener added with this method will report events on the call for as long as the implementation can listen to the JccCall. In the case that 1.the implementation can no longer observe the JccCall
2.this listener has been removed from this JccCall
3. the JccCall has completed and moved to the JccCall.INVALID state the application receives a CALL_EVENT_TRANSMISSION_ENDED event.

Multiple Invocations

Registering a single listener twice will result in the application being notified of the events specified in the filter implementation used in the last registration. Note that, because of this, registering for the same event multiple times should not result in multiple notifications being sent to an application for the same event.

Pre-conditions:

1. this.getState() != JcpCall.INVALID

Parameters:

`calllistener` - JcpCallListener object that receives the specified events.

`filter` - EventFilter which determines if the event is to be delivered to the specified listener.

Throws:

MethodNotSupportedException - The listener cannot be added at this time.

ResourceUnavailableException - The resource limit for the number of listeners has been exceeded.

---

addConnectionListener

```
public void addConnectionListener(JcpConnectionListener cl,
                                  EventFilter filter)
                           throws InvalidStateException
```

Add a connection listener to all connections under this call.

Registering a single listener twice will result in the last filter being used for the purposes of consultation to determine the events that the application is interested in. Note that registering for the same event multiple times should not result in multiple notifications being sent to an application for the same event.

Pre-conditions:

1. this.getState() != JcpCall.INVALID

Parameters:

`cl` - JcpConnectionListener object that receives the specified events.

`filter` - EventFilter determines if the event is to be delivered to the specified listener.

Throws:

InvalidStateException - If pre-conditions are not met.

---

removeConnectionListener

```
public void removeConnectionListener(JcpConnectionListener cl)
```

Removes the connection listener from all connections under this call. Note that if the listener is currently not registered then this method fails silently.

Parameters:

`cl` - JcpConnectionListener object that was registered using a corresponding addConnectionListener method.

---

release

```
public void release()
            throws PrivilegeViolationException,
                   ResourceUnavailableException,
                   InvalidStateException
```

This method requests the release of the call object and associated connection objects. Thus this method is equivalent to using the JccConnection.release() method on each JccConnection which is part of the Call. Typically each JccConnection associated with this call will move into the DISCONNECTED state. The call will also be terminated in the network. If the application has registered as a listener then it receives the CallEvent.CALL_EVENT_TRANSMISSION_ENDED event.

Pre-conditions:

1. (this.getProvider).getState == IN_SERVICE
2. this.getState ==ACTIVE

Post-conditions:

1. (this.getProvider).getState == IN_SERVICE
2. this.getState() == INVALID
3. JcpCallEvent.CALL_EVENT_TRANSMISSION_ENDED event delivered to the valid Calllisteners.
4. Appropriate ConnectionEvents are also delivered to the ConnectionListeners.

Throws:

PrivilegeViolationException - The application does not have the authority or permission to disconnect the Call. For example, an Address associated with this Call may not be controllable in the Provider's domain.

ResourceUnavailableException - An internal resource required to drop a connection is not available.

InvalidStateException - Some object required for the successful invocation of this method is not in the proper state as given by this method's pre-conditions.

---

createConnection

```
public JcpConnection createConnection(java.lang.String targetAddress,
                                      java.lang.String originatingAddress,
                                      java.lang.String originalCalledAddress,
                                      java.lang.String redirectingAddress)
                               throws InvalidStateException,
                                      ResourceUnavailableException,
                                      PrivilegeViolationException,
                                      MethodNotSupportedException
```

Creates a new JccConnection and attaches it to this JccCall. The JccConnection object is also associated with an JccAddress object corresponding to the targetAddress string given as an input parameter. Note that following this operation the JccConnection object might have to be routed to the JccAddress which can be accomplished using the JccConnection.routeConnection().

Pre-conditions:

1. this.getState() != JcpCall.INVALID
2. this.getProvider().getState ==JcpProvider.IN_SERVICE

Post-conditions:

1. let conn = createConnection(..);
2. conn.getState() == JcpConnection.IDLE state

3. this.getState() == JcpCall.ACTIVE
4. this.getProvider().getState ==JcpProvider.IN_SERVICE

Parameters:

`targetAddress` - specifies the JcpAddress with which the connection should be associated.

`originatingAddress` - specifies the address of the originating (calling) party. This is optional and can be set to null.

`originalCalledAddress` - specifies the initial address to which the call was initiated. This is optional and can be set to null.

`redirectingAddress` - specifies the last address from which the call was redirected. This is optional and can be set to null.

Returns:

JccConnection object created.

Throws:

InvalidStateException - Some object required by this method is not in a valid state as designated by the pre-conditions for this method.

ResourceUnavailableException - An internal resource necessary for creating the Connection object is unavailable.

PrivilegeViolationException - The application does not have the proper authority to create the Connection.

MethodNotSupportedException - The implementation does not support this method

---

routeCall

```
public JcpConnection routeCall(java.lang.String targetAddress,
                               java.lang.String originatingAddress,
                               java.lang.String originalDestinationAddress,
                               java.lang.String redirectingAddress)
                        throws InvalidStateException,
                               ResourceUnavailableException,
                               PrivilegeViolationException,
                               MethodNotSupportedException,
                               InvalidPartyException,
                               InvalidArgumentException
```

This method requests routing of a call to the given call party. This results in the creation of a JccConnection object associated with this JccCall. The JccConnection object is also associated with a JcpAddress passed as the parameter. Note that the address is passed as a string. The implementation is expected to find the JccAddress object corresponding to the string assuming that the JccAddress is local to the JcpProvider. The given string may not correspond to any JccAddress object in the JcpProvider's domain which would be the case for a call to a remote Address. This method is equivalent to the JccCall.createConnection(), JccConnection.routeConnection(FALSE)and JccConnection.attachMedia() or is also equivalent to JccCall.createConnection() and JccConnection.routeConnection(TRUE).

Pre-conditions:

1. this.getState() != JcpCall.INVALID
2. this.getProvider().getState == JcpProvider.IN_SERVICE

Post-conditions:

1. conn.getState() != IDLE or AUTHORISE_CALL_ATTEMPT where conn is the object returned as a result of this method.
2. this.getState() ==JcpCall.ACTIVE
3. this.getProvider().getState == JcpProvider.IN_SERVICE

Parameters:

`targetAddress` - specifies the origination party to which the call should be routed.

`originatingAddress` - specifies the address of the originating (calling) party. This parameter is optional and hence can be set to null.

`originalDestinationAddress` - specifies the original destination Address of the call. This parameter is optional and hence can be set to null.

Returns:

JcpConnection object created

Throws:

InvalidStateException - Some object required by this method is not in a valid state as designated by the pre-conditions for this method.

ResourceUnavailableException - An internal resource necessary for creating the Connection object is unavailable.

PrivilegeViolationException - The application does not have the proper authority to create the Connection.

MethodNotSupportedException - The implementation does not support this method

InvalidPartyException - The originator does not represent a valid party required to place a call.

InvalidArgumentException - The provided argument is not valid

---

superviseCall

```
public void superviseCall(JccCallListener calllistener,
                          double time,
                          int treatment,
                          double bytes)
                   throws MethodNotSupportedException
```

The application calls this method to supervise a call. The application can set a granted connection time for this call. If an application calls this function before it calls a routeCall(), the timer measurement will start as soon as the call is answered by the called party.
Note that a policy object may be designed to define the policy to be implemented by the platform as a result of this method instead of defining the policy through the given parameters. This might be designed in the future specifications.

Parameters:

`calllistener` - JccCallListener object that receives the specified events.

`time` - specifies the granted time in milliseconds for the connection. When specified as 0, volume based supervision is applied. Either bytes(volume) or time should be specified.

`treatment` - defines the treatment of the call by the call control service when the call supervision timer expires. The values which may be combined using a logical OR function are
1.01 to release the call when the call supervision timer expires.

2.02 to notify the application when the call supervision timer expires.
3. 04 to send a warning tone to the controlling party when a call supervision timer expires. If call release is requested, then the call will be released following the tone after an administered time period.

`bytes` - specifies the granted number of bytes that can be transmitted for the connection. When the quantity is specified as 0, time based supervision is applied.

Throws:

MethodNotSupportedException - if the implementation does not support this method.

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcc.JccCall

No usage of jain.jcc.JccCall

Overview Package Class Use Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Interface JcpCall

All Known Subinterfaces:

JccCall

public interface **JcpCall**

A JcpCall is a transient association of (zero or more) addresses for the purposes of engaging in a real-time communications interchange. The call and its associated connection and address objects describe the control and media flows taking place in some underlying "real world" communication network. Other parties involved in the call may also exert control over it, thus the membership and state of the endpoints may change without explicit request by the jcp application. The JcpProvider adjusts the call, address and connection objects to reflect the results of these combined command actions.

Introduction

A JcpCall can have zero or more `JcpConnection`s. A two-party call has two JcpConnections, and a conference call has three or more JcpConnections. Each JcpConnection models the relationship between a JcpCall and an JcpAddress, where an JcpAddress identifies a particular party or set of parties on a call.

Creating JcpCall Objects

Applications create instances of a JcpCall object with the `JcpProvider.createCall()` method, which returns a JcpCall object that has zero Connections and is in the `JcpCall.IDLE state`. The JcpCall maintains a reference to its JcpProvider for the life of that JcpCall object. The JcpProvider object instance does not change throughout the lifetime of the JcpCall object. The JcpProvider associated with a JcpCall is obtained via the `JcpCall.getProvider()` method.

JcpCall States

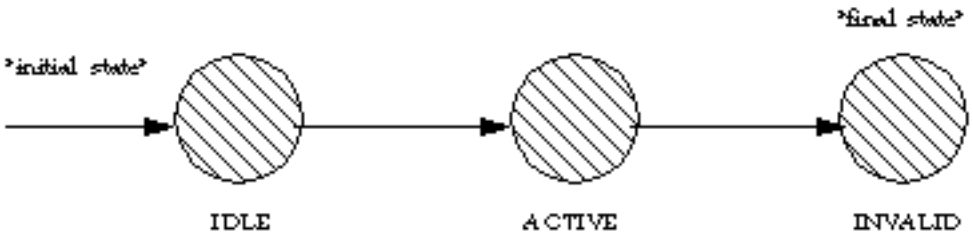
A JcpCall has a *state* which is obtained via the `JcpCall.getState()` method. This state describes the current progress of a telephone call, where is it in its life cycle, and how many connections exist on the call. The JcpCall state may be one of three values: `JcpCall.IDLE`, `JcpCall.ACTIVE`, or `JcpCall.INVALID`. The following is a description of each state:

`JcpCall.IDLE`	This is the initial state for all calls. In this state, the JcpCall has zero connections, that is `JcpCall.getConnections()` *must* return null.

`JcpCall.ACTIVE`	A call with some current ongoing activity is in this state. JcpCalls with one or more associated JcpConnections must be in this state. If a JcpCall is in this state, the `JcpCall.getConnections()` method must return an array of size at least one.
`JcpCall.INVALID`	This is the final state for all calls. JcpCall objects which lose all of their JcpConnection objects (via a transition of the JcpConnection object into the `JcpConnection.DISCONNECTED` state) moves into this state. Calls in this state have zero JcpConnections and these JcpCall objects may not be used for any future action. In this state, the `JcpCall.getConnections()` *must* return null.

JcpCall State Transitions

The possible Call state transitions are given in the diagram below:



JcpCall and JcpConnection objects

A JcpCall maintains a list of the JcpConnections on that JcpCall. Applications obtain an array of JcpConnections associated with the JcpCall via the `JcpCall.getConnections()` method. A JcpCall retains a reference to a JcpConnection only if it is not in the `JcpConnection.DISCONNECTED` state. Therefore, if a JcpCall has a reference to a JcpConnection, then that JcpConnection must not be in the `JcpConnection.DISCONNECTED` state. When a JcpConnection moves into the `JcpConnection.DISCONNECTED` state (e.g. when a party hangs up), the JcpCall loses its reference to that JcpConnection which is no longer reported via the `JcpCall.getConnections()` method.

Listeners and Events

The `JcpCallListener` interface reports all events pertaining to the JcpCall object. Events delivered to this interface must extend the `JcpCallEvent` interface. Applications can add listeners to a JcpCall object via the `JcpCall.addCallListener()` method.

Connection-related events can be reported via the `JcpCallListener` interface or via the `JcpConnectionListener` interface. These events include the creation of these objects and their state changes. Events which are reported via the `JcpCallListener` interface pertaining to JcpConnections extend the `JcpConnectionEvent` interface.

An event is delivered to the application whenever the state of the JcpCall changes. The event interfaces

corresponding to JcpCall state changes are: `CallActiveEventID` and `CallInvalidEventID`.

When Event Transmission Ends

At times it may become impossible for the implementation to report events to an application. In this case, a CALL_EVENT_TRANSMISSION_ENDED is delivered to an object registered as a `JcpCallListener` (or an extension of that interface).

This is the final event receives by the Listener.

Registering JcpCallListeners via Provider

Applications may receive events about a JcpCall by adding a Listener via the JcpCall or JcpProvider objects using the `addCallListener()` methods.

See Also:

`JcpCallListener`, `JcpConnectionListener`, `JcpConnection`, `JcpAddress`, `JcpCallEvent`

Field Summary

static int	**ACTIVE** JcpCall.ACTIVE state indicates the Call has one or more Connections none of which is in the JcpConnection.DISCONNECTED state.
static int	**IDLE** JcpCall.IDLE state indicates the Call has zero Connections.
static int	**INVALID** The JcpCall.INVALID state indicates that the Call has lost all of its connections, that is, all of its Connection objects have moved into the JcpConnection.DISCONNECTED state and are no longer associated with the Call.

Method Summary

void	**addCallListener**(JcpCallListener calllistener) Add a listener to this call.
JcpConnection[]	**getConnections**() Retrieves an array of connections asssociated with this call.
JcpProvider	**getProvider**() Retrieves the provider handling this call object.
int	**getState**() Retrieves the state of the call.

void	**removeCallListener**(JcpCallListener calllistener) Removes a listener from this call.

Field Detail

IDLE

```
public static final int IDLE
```

JcpCall.IDLE state indicates the Call has zero Connections. This is the initial state of all Call objects.

ACTIVE

```
public static final int ACTIVE
```

JcpCall.ACTIVE state indicates the Call has one or more Connections none of which is in the JcpConnection.DISCONNECTED state. The Call object transitions into this state from the IDLE state only.

INVALID

```
public static final int INVALID
```

The JcpCall.INVALID state indicates that the Call has lost all of its connections, that is, all of its Connection objects have moved into the JcpConnection.DISCONNECTED state and are no longer associated with the Call. A Call in this state cannot be used for future actions.

Method Detail

getState

```
public int getState()
```

Retrieves the state of the call. The state will be either IDLE, ACTIVE or INVALID.

Returns:

Integer representing the state of the call. See static int's defined in this object.

---

addCallListener

```
public void addCallListener(JcpCallListener calllistener)
                     throws ResourceUnavailableException,
                            MethodNotSupportedException
```

Add a listener to this call. This also reports all state changes in the state of the Call and Connection objects. The listener added with this method will report events on the call for as long as the implementation can listen to the Call. In the case that 1.the implementation can no longer observe the Call
2.this listener has been removed from this Call
3. the Call has completed and moved to the JcpCall.INVALID state the application receives a CALL_EVENT_TRANSMISSION_ENDED event.

Event Snapshots

By default, when an listener is added to a telephone call, the first batch of events may be a "snapshot". That is, if the listener was added after state changes in the Call, the first batch of events will inform the application of the current state of the Call. Note that these snapshot events do NOT provide a history of all events on the Call, rather they provide the minimum necessary information to bring the application up-to-date with the current state of the Call.

CallListeners from Provider

There may be additional call listeners on the call which were not added by this method. These listeners may have become part of the call via the `JcpProvider.addCallListener()` method. See the specifications for these methods for more information.

Multiple Invocations

If an application attempts to add an instance of an listener already present on this Call, there are two possible outcomes:

1. If the listener was added by the application using this method, then a repeated invocation will silently fail, i.e. multiple instances of an listener are not added and no exception will be thrown.
2. If the listener is part of the call because an application invoked `JccProvider.addCallListener()` either of these methods modifies the behavior of that listener as if it were added via this method instead.

Post-Conditions:

1. A snapshot of events is delivered to the listener, if appropriate.

Parameters:

`calllistener` - JcpCallListener object that receives the specified events.

Throws:

MethodNotSupportedException - The listener cannot be added at this time.

ResourceUnavailableException - The resource limit for the number of listeners has been exceeded.

---

removeCallListener

`public void` **`removeCallListener`**`(JcpCallListener calllistener)`

Removes a listener from this call. If successful, the listener will receive a CALL_EVENT_TRANSMISSION_ENDED as the last event it receives. If the listener is not part of the Call for the given address(es), then this method fails silently, i.e. no listener is removed and no exception is thrown.

This method has different effects depending upon how the listener was added to the Call, as follows:

1. If the listener was added via `JcpCall.addCallListener()`, this method removes the listener until it is re-applied by the application.
2. If the listener was added via `JccProvider.addCallListener()`, this method removes the listener for this call only. It does not affect whether this listener will be added to future calls which come to that Address. See `JccProvider.addCallListener()` for more details.

If an listener is not part of the Call, then this method fails silently, i.e. no listener is removed and no exception is thrown.

Post-Conditions:

1. CALL_EVENT_TRANSMISSION_ENDED is delivered to the application

Parameters:

`calllistener` - JcpCall Listener object.

---

getProvider

`public JcpProvider` **`getProvider`**`()`

Retrieves the provider handling this call object. The Provider reference does not change once the Call object has been created, despite the state of the Call object.

Returns:

JcpProvider object managing this call.

---

getConnections

`public JcpConnection[] getConnections()`

Retrieves an array of connections asssociated with this call. None of the Connections returned will be in the JcpConnection.DISCONNECTED state. Further, if the Call is in the IDLE or INVALID state, this method returns null.

Post-Conditions:

1. JcpConnection[] conn=JcpCall.getConnections()
2. if this.getState() == JcpCall.IDLE then conn=null
3.if this.getState == JcpCall.INVALID then conn=null
4. if this.getState == JcpCall.ACTIVE then conn.length >=1
5. For all i, conn[i].getState() != JcpConnection.DISCONNECTED

Returns:

Array of Connections for this call.

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcp.JcpCall

Packages that use JcpCall

jain.jcc	
jain.jcp	

Uses of JcpCall in jain.jcc

Subinterfaces of JcpCall in jain.jcc	
interface	**JccCall** The JccCall interface extends the JcpCall interface of JCP.

Uses of JcpCall in jain.jcp

Methods in jain.jcp that return JcpCall	
JcpCall	**JcpProvider.createCall()** Creates a new instance of the call with no connections.
JcpCall	**JcpConnection.getCall()** Retrieves the Jcpcall that is associated with this Jcpconnection.
JcpCall	**JcpCallEvent.getCall()** Returns the JcpCall object associated with this event.

jain.jcp

Interface JcpConnection

All Known Subinterfaces:

JccConnection

public interface **JcpConnection**

Introduction

The purpose of a JcpConnection object is to describe the relationship between a JcpCall object and a JcpAddress object. A JcpConnection object exists if the JcpAddress is a part of the telephone call. Each JcpConnection has a *state* which describes the particular stage of the relationship between the JcpCall and JcpAddress. These states and their meanings are described below. Applications use the `JcpConnection.getCall()` and `JcpConnection.getAddress()` methods to obtain the JcpCall and JcpAddress associated with this JcpConnection, respectively.

From one perspective, an application may view a JcpCall only in terms of the JcpAddress/JcpConnection objects which are part of the JcpCall. This is termed a *logical* view of the Call. In this logical view, a telephone call is viewed as two or more endpoint addresses in communication. The JcpConnection object describes the state of each of these endpoint addresses with respect to the JcpCall.

JcpCalls and JcpAddresses

JcpConnection objects are immutable in terms of their JcpCall and JcpAddress references. In other words, the JcpCall and JcpAddress object references do not change throughout the lifetime of the JcpConnection object instance. The same JcpConnection object may not be used in another telephone call. The existence of a JcpConnection implies that its JcpAddress is associated with its JcpCall in the manner described by the JcpConnection's state.

Although a JcpConnection's JcpAddress and JcpCall references remain valid throughout the lifetime of the JcpConnection object, the same is not true for the JcpCall and JcpAddress object's references to this JcpConnection. Particularly, when a JcpConnection moves into the JcpConnection.DISCONNECTED state, it is no longer listed by the `JcpCall.getConnections()` method. Typically, when a JcpConnection moves into the `JcpConnection.DISCONNECTED` state, the application loses its references to it to facilitate its garbage collection.

Connection States

Below is a description of each JcpConnection state in real-world terms. These real-world descriptions have no bearing on the specifications of methods, they only serve to provide a more intuitive

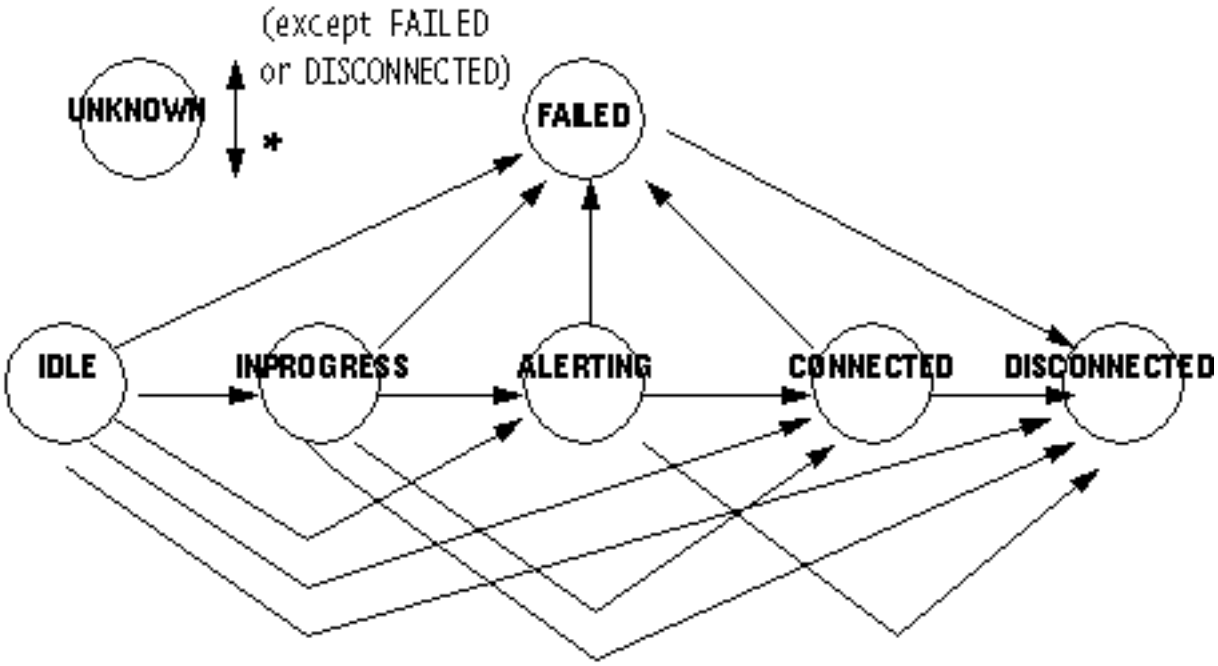
understanding of what is going on. Several methods in this specification state pre-conditions based upon the state of the Connection.

`JcpConnection.IDLE`	This state is the initial state for all new Connections. Connections which are in the `JcpConnection.IDLE` state are not actively part of a telephone call, yet their references to the Call and Address objects are valid. Connections typically do not stay in the `JcpConnection.IDLE` state for long, quickly transitioning to other states.
`JcpConnection.DISCONNECTED`	This state implies it is no longer part of the telephone call, although its references to Call and Address still remain valid. A Connection in this state is interpreted as once previously belonging to this telephone call.
`JcpConnection.INPROGRESS`	This state implies that the Connection, which represents the destination end of a telephone call, is in the process of contacting the destination side. Under certain circumstances, the Connection may not progress beyond this state. Extension packages elaborate further on this state in various situations.
`JcpConnection.ALERTING`	This state implies that the Address is being notified of an incoming call.
`JcpConnection.CONNECTED`	This state implies that a Connection and its Address is actively part of a telephone call. In common terms, two people talking to one another are represented by two Connections in the `JcpConnection.CONNECTED` state.
`JcpConnection.UNKNOWN`	This state implies that the implementation is unable to determine the current state of the Connection. Typically, methods are invalid on Connections which are in this state. Connections may move in and out of the `JcpConnection.UNKNOWN` state at any time.
`JcpConnection.FAILED`	This state indicates that a Connection to that end of the call has failed for some reason. One reason why a Connection would be in the `JcpConnection.FAILED` state is because the party was busy.

Connection State Transitions

With these loose, real-world meanings in the back of one's mind, the JcpConnection class defines a finite-state diagram which describes the allowable JcpConnection state transitions. This finite-state diagram must be guaranteed by the implementation. Each method which causes a change in a JcpConnection state must be consistent with this state diagram. This finite state diagram is below:

Note there is a general left-to-right progression of the state transitions. A JcpConnection object may transition into and out of the `JcpConnection.UNKNOWN` state at any time (hence, the asterisk qualifier next to its bidirectional transition arrow).



Listeners and Events

All events pertaining to the JcpConnection object are reported via the `JcpCallListener` interface on the JcpCall object associated with this JcpConnection. Events are reported to a JcpCallListener when a new JcpConnection is created and whenever a JcpConnection changes state. Listeners are added to JcpCall objects via the `JccCall.addCallListener()` method and more indirectly via the `JccProvider.addCallListener()` method. See the specifications for the JccCall and JccProvider interfaces for more information.

Field Summary

`static int`	**`ALERTING`** The `JcpConnection.ALERTING` state implies that the Address is being notified of an incoming call.
`static int`	**`CONNECTED`** The `JcpConnection.CONNECTED` state implies that a Connection and its Address is actively part of a telephone call.
`static int`	**`DISCONNECTED`** The `JcpConnection.DISCONNECTED` state implies it is no longer part of the telephone call, although its references to Call and Address still remain valid.
`static int`	**`FAILED`** The `JcpConnection.FAILED` state indicates that a Connection to that end of the call has failed for some reason.

static int	**IDLE** The JcpConnection.IDLE state is the initial state for all new Connections.
static int	**INPROGRESS** The JcpConnection.INPROGRESS state implies that the Connection, which represents the destination end of a telephone call, is in the process of contacting the destination side.
static int	**UNKNOWN** The JcpConnection.UNKNOWN state implies that the implementation is unable to determine the current state of the Connection.

Method Summary

JcpAddress	**getAddress**() Returns the JcpAddress associated with this JcpConnection.
JcpCall	**getCall**() Retrieves the Jcpcall that is associated with this Jcpconnection.
int	**getState**() Retrieves the state of the JcpConnection object.

Method Detail

getState

public int **getState**()

Retrieves the state of the JcpConnection object.

Returns:

Integer representing the state of the call. See static int's defined in this object.

getCall

public JcpCall **getCall**()

Retrieves the Jcpcall that is associated with this Jcpconnection. This JcpCall reference remains valid throughout the lifetime of the JcpConnection object despite the state of the JcpConnection object. This JcpCall reference does not change once the JcpConnection object has been created.

Returns:

JcpCall object holding this connection.

---

getAddress

`public JcpAddress` **`getAddress`**`()`

Returns the JcpAddress associated with this JcpConnection. This JcpAddress object remains valid throughout the lifetime of the JcpConnection object despite the state of the JcpConnection object. This JcpAddress reference does not change once the JcpConnection object has been created.

Returns:

JcpAddress object associated with this JcpConnection object.

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcp.JcpConnection

Packages that use JcpConnection

jain.jcc	
jain.jcp	

Uses of JcpConnection in jain.jcc

Subinterfaces of JcpConnection in jain.jcc	
interface	**JccConnection** A JccConnection object represents a link between a network endpoint (address) and a JccCall object.

Methods in jain.jcc that return JcpConnection	
JcpConnection	**JccCall.createConnection**(java.lang.String targetAddress, java.lang.String originatingAddress, java.lang.String originalCalledAddress, java.lang.String redirectingAddress) Creates a new JccConnection and attaches it to this JccCall.
JcpConnection	**JccCall.routeCall**(java.lang.String targetAddress, java.lang.String originatingAddress, java.lang.String originalDestinationAddress, java.lang.String redirectingAddress) This method requests routing of a call to the given call party.

Uses of JcpConnection in jain.jcp

Methods in jain.jcp that return JcpConnection

JcpConnection[]	**JcpCall.getConnections()** Retrieves an array of connections asssociated with this call.
JcpConnection	**JcpConnectionEvent.getConnection()** Returns the JcpConnection associated with this event.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcc

Interface JccConnection

public interface **JccConnection**

extends JcpConnection

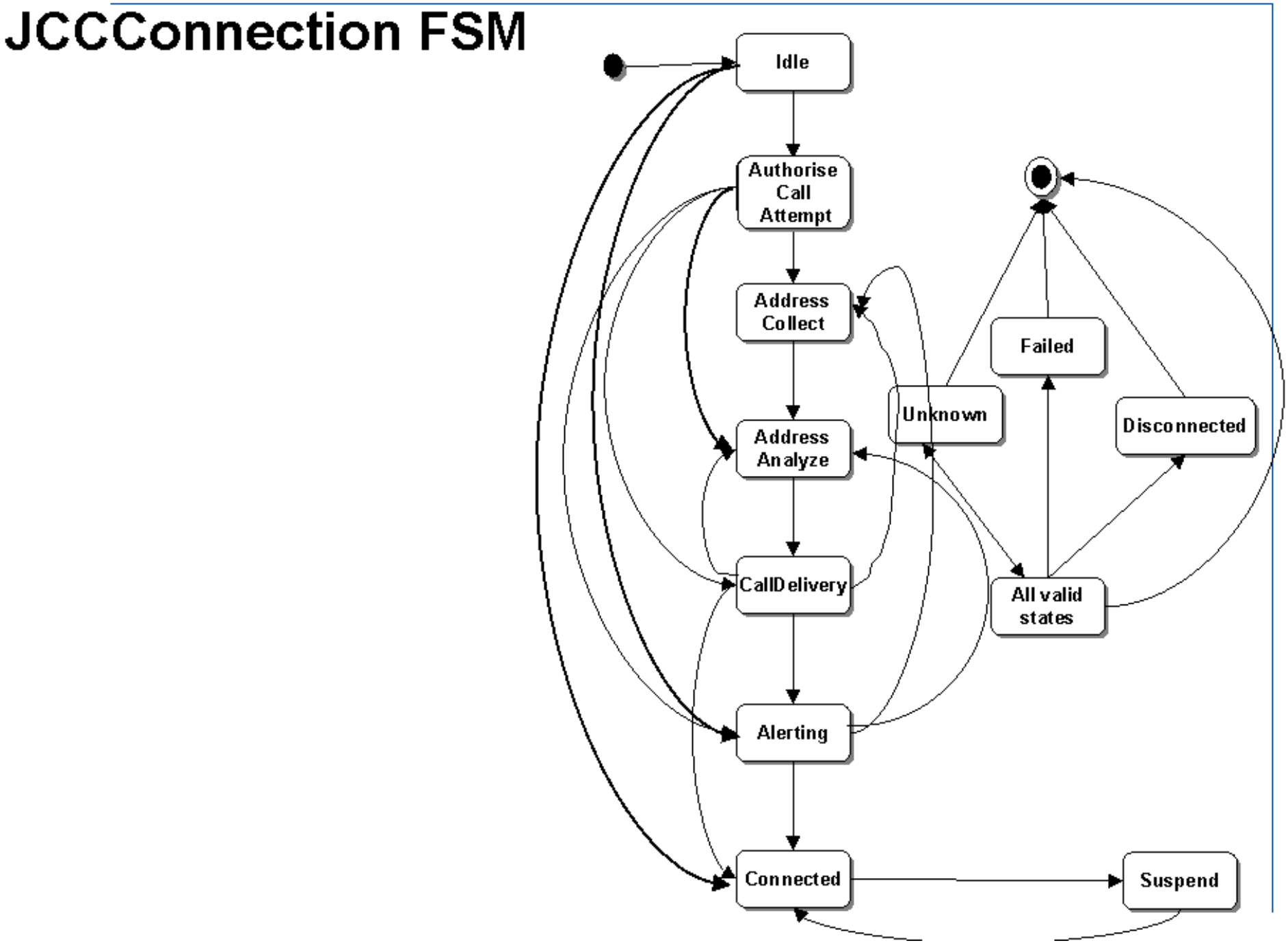
A JccConnection object represents a link between a network endpoint (address) and a JccCall object.

Jcp vs. Jcc Package States

There is a strong relationship between the `JccConnection` states and the `JcpConnection` states. If an implementation supports the JCC package, it must ensure this relationship is properly maintained.

JccConnection State Transitions

The JccConnection class defines a finite-state diagram which describes the allowable JccConnection state transitions. This finite-state diagram must be guaranteed by the implementation. Each method which causes a change in a JccConnection state must be consistent with this state diagram. This finite state diagram is below:



Since the states defined in the `JccConnection` interface provide more detail to the states defined in the `JcpConnection` interface, each state in the

`JcpConnection` interface corresponds to a state defined in the `JccConnection` interface. Conversely, each `JccConnection` state corresponds to exactly one `JcpConnection` state. This arrangement permits applications to view either the core state or the JCC state and still see a consistent view.

The following table outlines the relationship between the JCP package Connection states and the JCC package Connection states.

If the JCC package state is...	*then the JCP state must be...*
`JccConnection.IDLE`	`JcpConnection.IDLE`
`JccConnection.AUTHORIZE_CALL_ATTEMPT`	`JcpConnection.INPROGRESS`
`JccConnection.ADDRESS_COLLECT`	`JcpConnection.INPROGRESS`
`JccConnection.ADDRESS_ADDRESS_ANALYZE`	`JcpConnection.INPROGRESS`
`JccConnection.CALL_DELIVERY`	`JcpConnection.INPROGRESS`
`JccConnection.ALERTING`	`JcpConnection.ALERTING`
`JccConnection.CONNECTED`	`JcpConnection.CONNECTED`
`JccConnection.SUSPENDED`	`JcpConnection.CONNECTED`
`JccConnection.DISCONNECTED`	`JcpConnection.DISCONNECTED`
`JccConnection.FAILED`	`JcpConnection.FAILED`
`JccConnection.UNKNOWN`	`JcpConnection.UNKNOWN`

Events--Blocking and non-Blocking

All the events on the JccConnection are expected to be blockable. In other words, after sending each event to the listener the implementation can either suspend processing or continue with processing. The implementation suspends processing if the event is to be fired in a blocking mode and the implementation continues with processing if the event is to be fired in a non-blocking mode. In case of a blocking event, the implementation is expected to suspend processing either until the application uses a valid API method call or until a timeout occurs.

The listeners are expected to specify the mode in which they are to be notified of the events. Note that the events are sent out only when the state is reached. Hence, when processing is suspended the connection is in some state.

Field Summary

static int	**ADDRESS_ANALYZE** Represents the connection ADDRESS_ANALYZE state.
static int	**ADDRESS_COLLECT** Represents the connection ADDRESS_COLLECT state.
static int	**ALERTING** Represents the connection ALERTING state.
static int	**AUTHORIZE_CALL_ATTEMPT** Represents the connection AUTHORIZE_CALL_ATTEMPT state.
static int	**CALL_DELIVERY** Represents the connection CALL_DELIVERY state.
static int	**CONNECTED** Represents the connection CONNECTED state.

static int	**DISCONNECTED** Represents the connection DISCONNECTED state.
static int	**FAILED** Represents the FAILED state.
static int	**IDLE** Represents the connection IDLE state.
static int	**SUSPENDED** Represents the SUSPEND state.
static int	**UNKNOWN** Represents the UNKNOWN state.

Method Summary

void	**answer**() This method causes the call to be answered.
void	**attachMedia**() This method will allow transmission on all associated bearer connections or media channels to and from other parties in the call.
void	**continueProcessing**() This method requests the platform to continue processing.
void	**detachMedia**() This method will detach the JccConnection from the call, i.e., this will prevent transmission on any associated bearer connections or media channels to and from other parties in the call.
int	**getJccState**() Retrieves the state of the JccConnection object.
JcpAddress	**getLastAddr**() Returns the last redirected JcpAddress associated with this JcpCall.
java.lang.String	**getMoreDialledDigits**() This method is used by the application to instruct the platform to collect further digits and return them to the application.
JcpAddress	**getOriginalAddress**() Returns the original JcpAddress associated with this JcpCall.
boolean	**isBlocked**() Returns a boolean value indicating if the JccConnection is currently blocked due to a blocking event having been fired to a listener registered for that blocking event.
void	**release**() Drops a JccConnection from an active telephone call.

void	**routeConnection**(boolean attachmedia) Routes this JccConnection to the specified target address.

Methods inherited from interface jain.jcp.JcpConnection
getAddress, getCall, getState

Field Detail

DISCONNECTED

public static final int **DISCONNECTED**

Represents the connection DISCONNECTED state. This state implies that the JccConnection object is disconnected from the call. *Entry criteria:* This state is entered when a disconnect indication is received from the corresponding party or the application. *Function:* The connections for the originating and terminating party are disconnected and depending on the incoming network connection, appropriate backward signaling takes place. *Exit criteria:*

IDLE

public static final int **IDLE**

Represents the connection IDLE state. This state is the initial state for all new JccConnection objects. A JccConnection object in the IDLE state while not yet actively participating in a call can still reference a JccCall and JccAddress object.
Entry criteria Start of a new call.
Functions: Interface (line/trunk) is idled.
Exit criteria: An indication of the desire to place an outgoing call or when the indication of an incoming call is received. In both cases the respective connections move to the AUTHORIZE_CALL_ATTEMPT state.

AUTHORIZE_CALL_ATTEMPT

public static final int **AUTHORIZE_CALL_ATTEMPT**

Represents the connection AUTHORIZE_CALL_ATTEMPT state. This state implies that the originating or terminating terminal needs to be authorized for the call.
Entry criteria An indication that the originating or terminating terminal needs to be authorized for the call.
Functions: The originating or terminating terminal characteristics should be verified using the calling party's identity and service profile. The authority/ability of the party to place the call with given properties is verified. The types of authorization may vary for different types of originating and terminating resources.
Exit criteria: The JccConnection object exits this state on receiving indication of the success or failure of the authorization process. The originating JccConnection might move to the ADDRESS_COLLECT state while the terminating JccConnection has to move to the CALL_DELIVERY state or beyond. Thus, the terminating JccConnection cannot be either in the ADDRESS_COLLECT or the ADDRESS_ANALYZE states.

ADDRESS_COLLECT

`public static final int` **`ADDRESS_COLLECT`**

Represents the connection ADDRESS_COLLECT state.
Entry criteria The JccConnection object enters this state with the originating party having been authorized for this call.
Functions: In this state the initial information package is collected from the originating party. Information is examined according to dialing plan to determine the end of collection. No further action may be required if en bloc signaling method is in use.
Exit criteria: This state is exited either because the complete initial information package or dialing string has been collected from the originating party or because of failure to collect information or even due to reception of invalid information from the caller. Timeout and abandon indications may also cause the exit from this state.

ADDRESS_ANALYZE

`public static final int` **`ADDRESS_ANALYZE`**

Represents the connection ADDRESS_ANALYZE state.

Entry criteria This state is entered on the availability of complete initial information package/dialing string from the originating party.
Functions: The information collected is analyzed and/or translated according to a dialing plan to determine routing address and call type (e.g. local exchange call, transit exchange call, international exchange call).
Exit criteria: This state is exited on the availability of routing address. Invalid information and Abandon indications also cause transition out of this state. Exception criteria such as network busy, abandon, route busy etc. will cause exit from this state.

CALL_DELIVERY

`public static final int` **`CALL_DELIVERY`**

Represents the connection CALL_DELIVERY state. *Entry criteria:* This state is entered on the originating side when the routing address and call type are available. On the terminating side this state is entered when the termination attempt to the address is authorized. *Function:* On the originating side this state involves selecting of the route as well as sending an indication of the desire to set up a call to the specified called party. On the terminating side this state is involves checking the busy/idle status of the terminating access and also informing the terminating message of an incoming call. *Exit criteria:* This state is exited on the originating side when criteria such as receipt of an alerting indication or call accepted is received from the terminating call portion. This state is exited on the terminating side when the terminating party is being alerted or the call is accepted.

ALERTING

`public static final int` **`ALERTING`**

Represents the connection ALERTING state. This state implies that the address object is being notified of an incoming call.
Entry criteria: This state is entered when the terminating party is being alerted of an incoming call. *Function:* An indication is sent to the originating party that the terminating party is being alerted. *Exit criteria:* This state is exited when the call is accepted and answered by the terminating party. Exception criteria such as

callrejected, NoAnswer and Abandon all cause exit from this state.

CONNECTED

```
public static final int CONNECTED
```

Represents the connection CONNECTED state. This state implies that an originating and terminating connection objects and the associated Address objects are actively part of a call. *Entry criteria:* This state is entered when the Call is accepted and answered by the terminating party. *Function:* In this state several processes related to message accounting/charging, call supervision etc. may be initiated. *Exit criteria:* Exception criteria such as disconnect and suspend cause exit from this state.

SUSPENDED

```
public static final int SUSPENDED
```

Represents the SUSPEND state. This state implies that this JccConnection object is suspended from the call, although it's references to a JccCall and JccAddress objects will stil remain valid. *Entry criteria:* A suspend indication is received that the terminating party has disconnected, but disconnect timing has not completed. This state might also be entered on cases like the flash hook. *Function:* The connections for the originating and terminating party are maintained and depending on the incoming network connection, appropriate backward signaling takes place. *Exit criteria:* Exception criteria such as disconnect cause exit from this state.

FAILED

```
public static final int FAILED
```

Represents the FAILED state. This indicates that a JccConnection to that end of the call has failed for some reason. One reason why a JccConnection would be in the FAILED state is due to the fact that the party was busy. *Entry criteria:* This state is entered when an exception condition is encountered. *Function:* Default handling of the exception condition is provided. *Exit criteria:* Default handling of the exception condition by the JCC implementation is completed.

UNKNOWN

```
public static final int UNKNOWN
```

Represents the UNKNOWN state. This indicates that the platform does not know of the current state of the corresponding JccConnection object.

Method Detail

getJccState

```
public int getJccState()
```

Retrieves the state of the JccConnection object.

Returns:

Integer representing the state of the call. See static int's defined in this object.

---

routeConnection

```
public void routeConnection(boolean attachmedia)
                    throws InvalidStateException,
                           ResourceUnavailableException,
                           PrivilegeViolationException,
                           MethodNotSupportedException,
                           InvalidPartyException,
                           InvalidArgumentException
```

Routes this JccConnection to the specified target address.

Pre-Conditions:

1. this.getState() == JccConnection.IDLE or JccConnection.AUTHORISE_CALL_ATTEMPT

Post-Conditions:

1. this.getJccState() != IDLE OR AUTHORISE_CALL_ATTEMPT

Note that this JccConnection may have progressed beyond the ADDRESS_COLLECTED state.

Parameters:

`attachmedia` - indicates if the media has to be attached after the connection is routed. TRUE causes the media to be attached, FALSE causes the media not to be attached in which case a separate call to attachMedia() must be made in order to attach the media to this connection.

Throws:

InvalidStateException - Some object required by this method is not in a valid state as designated by the pre-conditions for this method.

ResourceUnavailableException - An internal resource for completing this call is unavailable.

PrivilegeViolationException - The application does not have the proper authority to call this method.

MethodNotSupportedException - The implementation does not support this method.

InvalidPartyException - The given Addresses are not valid.

InvalidArgumentException - The provided argument is not valid.

---

release

```
public void release()
            throws PrivilegeViolationException,
                   ResourceUnavailableException,
```

InvalidStateException

Drops a JccConnection from an active telephone call. If successful, the associated JccAddress will be released from the call and the JccConnection moves to the DISCONNECTED state following which it may be deleted. The JccConnection's JccAddress is no longer associated with the telephone call. This method does not necessarily drop the entire telephone call, only the particular JccConnection on the telephone call. This method provides the ability to disconnect a specific party from a telephone call, which is especially useful in telephone calls consisting of three or more parties. Invoking this method may result in the entire telephone call being dropped, which is a permitted outcome of this method. In that case, the appropriate events are delivered to the application, indicating that more than just a single JccConnection has been dropped from the telephone call. As a result of this method returning successfully, a JccConnectionDisconnected event for this JccConnection is delivered to the registered listeners.

Dropping Additional Connections

Additional JccConnections may be dropped indirectly as a result of this method. For example, dropping the destination JccConnection of a two-party call may result in the entire telephone call being dropped. It is up to the implementation to determine which JccConnections are dropped as a result of this method. Implementations should not, however, drop additional JccConnections representing additional parties if it does not reflect the natural response of the underlying telephone hardware.

Pre-conditions:

1. this.getState() != JccConnection.IDLE or JccConnection.DISCONNECTED
2. ((this.getCall()).getProvider()).getState() == JccProvider.IN_SERVICE
3. (this.getCall()).getState() ==JccCall.ACTIVE

Post-conditions:

1. this.getState() == JccConnection.DISCONNECTED
2. ((this.getCall()).getProvider()).getState() == JccProvider.IN_SERVICE
3. Connection_Disconnected event is delivered for to the registered listeners.
4. CallInvalid event is also delivered if all the JccConnections are dropped indirectly as a result of this method.

Throws:

InvalidStateException - If either of the JccConnection, JccCall or JccProvider objects is not in the proper states as given by this method's precondition.

PrivilegeViolationException - The application does not have the authority or permission to disconnect the JccConnection. For example, the JccAddress associated with this JccConnection may not be controllable in the JccProvider's domain.

ResourceUnavailableException - An internal resource to drop a connection is not available.

---

answer

```
public void answer()
```

This method causes the call to be answered.

Pre-conditions:

1. this.getState() != JccConnection.CONNECTED or JccConnection.DISCONNECTED or JccConnection.FAILED
2. ((this.getCall()).getProvider()).getState() == JccProvider.IN_SERVICE
3. (this.getCall()).getState() ==JccCall.ACTIVE

Post-conditions:

1. this.getState() == JccConnection.CONNECTED
2. ((this.getCall()).getProvider()).getState() == JccProvider.IN_SERVICE
3. JccConnectionConnected event is delivered for to the registered listeners.
4. (this.getCall()).getState() ==JccCall.ACTIVE

---

continueProcessing

`public void` **`continueProcessing`**`()`

This method requests the platform to continue processing. The call processing has been suspended due to the firing of a blocking event (trigger) and this method causes the processing to continue.

---

attachMedia

`public void` **`attachMedia`**`()`

This method will allow transmission on all associated bearer connections or media channels to and from other parties in the call. The JccConnection object must be in the CONNECTED state for this method to complete successfully.

---

detachMedia

`public void` **`detachMedia`**`()`

This method will detach the JccConnection from the call, i.e., this will prevent transmission on any associated bearer connections or media channels to and from other parties in the call. The JccConnection object must be in the CONNECTED state for this method to complete successfully.

---

isBlocked

`public boolean` **`isBlocked`**`()`

Returns a boolean value indicating if the JccConnection is currently blocked due to a blocking event having been fired to a listener registered for that blocking event. The method returns false once a valid API call is made after the firing of a blocking event or until after the expiry of a timeout.

Returns:

boolean indicating if the connection is blocked due to a blocking event.

---

getMoreDialledDigits

`public java.lang.String` **`getMoreDialledDigits`**`()`

This method is used by the application to instruct the platform to collect further digits and return them to the application. The platform is then expected to return the collected digits as a String in using

---

getLastAddr

`public` JcpAddress **`getLastAddr`**`()`

Returns the last redirected JcpAddress associated with this JcpCall. The last redirected JcpAddress is the JcpAddress at which the current JcpCall was placed immediately before the current JcpAddress. This is common if a JcpCall is forwarded to several JcpAddresses before being answered. If the last redirected address is unknown or not yet known, this method returns null.

Returns:

the JcpAddress to which the call was last associated and redirection on which caused the current Address to be associated with the call through the connection.

---

getOriginalAddress

`public` JcpAddress **`getOriginalAddress`**`()`

Returns the original JcpAddress associated with this JcpCall. This would be the first JcpAddress to which the call was placed. The current JcpAddress might be different from this due to multiple forwardings. If this JcpAddress is unknown or not yet known, this method returns null.

Returns:

the JcpAddress which was called initially.

Uses of Interface jain.jcc.JccConnection

No usage of jain.jcc.JccConnection

Overview Package Class Use Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcc

Interface JccCallListener

All Known Subinterfaces:

JccConnectionListener

public interface **JccCallListener**
extends JcpCallListener

This interface reports all changes to the JccCall object.

The `JccConnectionListener` interface extends this interface. This reflects the fact that all JccConnection events can be reported via the `JccCallListener` interface.

Method Summary

void	**callSuperviseEnd**(JccCallEvent callevent) Indicates that the supervision of the call has ended.
void	**callSuperviseStart**(JccCallEvent callevent) Indicates that the supervision of the call has started.

Methods inherited from interface jain.jcp.JcpCallListener

callActive, callcreated, callEventTransmissionEnded, callInvalid

Method Detail

callSuperviseStart

```
public void callSuperviseStart(JccCallEvent callevent)
```

Indicates that the supervision of the call has started.

Parameters:

`callevent` - JccCallevent.

callSuperviseEnd

`public void` **`callSuperviseEnd`**`(JccCallEvent callevent)`

Indicates that the supervision of the call has ended.

Parameters:

`callevent` - JccCallevent.

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcc.JccCallListener

Packages that use JccCallListener

jain.jcc

Uses of JccCallListener in jain.jcc

Subinterfaces of JccCallListener in jain.jcc	
interface	**JccConnectionListener** This interface is an extension of the JccCallListener and the JcpConnectionListener interface and reports state changes both of the JccCall and its JccConnections.

Methods in jain.jcc with parameters of type JccCallListener	
void	**JccCall.superviseCall**(JccCallListener calllistener, double time, int treatment, double bytes) The application calls this method to supervise a call.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Interface JcpCallListener

All Known Subinterfaces:

JccCallListener, JccConnectionListener, JcpConnectionListener

public interface **JcpCallListener**

extends java.util.EventListener

This interface reports all changes to the Call object. The `JcpCallEvent` interface is the base interface for all Call-related events. All Call-related events must extend this interface. Events which extend this interface are reported via the `JcpCallListener` interface.

An individual `JcpCallEvent` conveys one of a series of different possible Call state changes; the specific Call state change is indicated by the `Event.getID()` value returned by the event.

The `JcpConnectionEvent` interface extends this interface. This reflects the fact that all Connection events can be reported via the `JcpCallListener` interface.

The `JcpCallEvent.getCall()` method on this interface returns the Call associated with the Call event.

See Also:

`Event`, `JcpConnectionEvent`, `JcpCallListener`, `JcpCall`

Method Summary

void	**`callActive`**`(JcpCallEvent callevent)` Indicates that the state of the JcpCall object has changed to JcpCall.ACTIVE.
void	**`callcreated`**`(JcpCallEvent callevent)` Indicates that the state of the JcpCall object has changed to JcpCall.IDLE.
void	**`callEventTransmissionEnded`**`(JcpCallEvent callevent)` This method is called to indicate that the application will no longer receive JcpCallEvent events on the instance of the JcpCallListener.
void	**`callInvalid`**`(JcpCallEvent callevent)` Indicates that the state of the JcpCall object has changed to JcpCall.INVALID.

Method Detail

callActive

`public void` **`callActive`**`(JcpCallEvent callevent)`

Indicates that the state of the JcpCall object has changed to JcpCall.ACTIVE.

Parameters:

`callevent` - JcpCallevent with eventID CALL_ACTIVE.

---

callInvalid

`public void` **`callInvalid`**`(JcpCallEvent callevent)`

Indicates that the state of the JcpCall object has changed to JcpCall.INVALID.

Parameters:

`callevent` - JcpCallevent with eventID CALL_INVALID.

---

callEventTransmissionEnded

`public void` **`callEventTransmissionEnded`**`(JcpCallEvent callevent)`

This method is called to indicate that the application will no longer receive JcpCallEvent events on the instance of the JcpCallListener.

Parameters:

`callevent` - JcpCallevent with eventID CALL_EVENT_TRANSMISSION_ENDED.

---

callcreated

`public void` **`callcreated`**`(JcpCallEvent callevent)`

Indicates that the state of the JcpCall object has changed to JcpCall.IDLE.

Parameters:

`callevent` - JcpCallevent with eventID CALL_CREATED.

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcp.JcpCallListener

Packages that use JcpCallListener

jain.jcc	
jain.jcp	

Uses of JcpCallListener in jain.jcc

Subinterfaces of JcpCallListener in jain.jcc	
interface	**JccCallListener** This interface reports all changes to the JccCall object.
interface	**JccConnectionListener** This interface is an extension of the JccCallListener and the JcpConnectionListener interface and reports state changes both of the JccCall and its JccConnections.

Methods in jain.jcc with parameters of type JcpCallListener	
void	**JccProvider.addCallListener**(JcpCallListener calllistener) Add a call listener to all call objects that will be created under the domain of this provider.
void	**JccProvider.addCallListener**(JcpCallListener calllistener, EventFilter filter) Add a call listener to all call objects that will be created under the domain of this provider.
void	**JccProvider.removeCallListener**(JcpCallListener calllistener) Removes a call listener that was registered using JccProvider.addCallListener.
void	**JccCall.addCallListener**(JcpCallListener calllistener, EventFilter filter) Add a listener to this call.

Uses of JcpCallListener in jain.jcp

Subinterfaces of JcpCallListener in jain.jcp	
interface	**JcpConnectionListener** This interface is an extension of the JcpCallListener interface and reports state changes both of the JcpCall and its JcpConnections.

Methods in jain.jcp with parameters of type JcpCallListener	
void	**JcpCall.addCallListener**(JcpCallListener calllistener) Add a listener to this call.
void	**JcpCall.removeCallListener**(JcpCallListener calllistener) Removes a listener from this call.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Interface JcpConnectionListener

All Known Subinterfaces:

JccConnectionListener

public interface **JcpConnectionListener**

extends JcpCallListener

This interface is an extension of the JcpCallListener interface and reports state changes both of the JcpCall and its JcpConnections.

Method Summary

void	**connectionAlerting**(JcpConnectionEvent connectionevent) Indicates that the JcpConnection has just been placed in the JcpConnection.ALERTING state
void	**connectionConnected**(JcpConnectionEvent connectionevent) Indicates that the JcpConnection has just been placed in the JcpConnection.CONNECTED state
void	**connectionCreated**(JcpConnectionEvent connectionevent) Indicates that the JcpConnection object has just been created.
void	**connectionDisconnected**(JcpConnectionEvent connectionevent) Indicates that the JcpConnection has just been placed in the JcpConnection.DISCONNECTED state
void	**connectionFailed**(JcpConnectionEvent connectionevent) Indicates that the JcpConnection has just been placed in the JcpConnection.FAILED state
void	**connectionInProgress**(JcpConnectionEvent connectionevent) Indicates that the JcpConnection has just been placed in the JcpConnection.INPROGRESS state
void	**connectionUnknown**(JcpConnectionEvent connectionevent) Indicates that the Connection has just been placed in the JcpConnection.UNKNOWN state

Methods inherited from interface jain.jcp.JcpCallListener
callActive, callcreated, callEventTransmissionEnded, callInvalid

Method Detail

connectionCreated

`public void` **`connectionCreated`**`(JcpConnectionEvent connectionevent)`

Indicates that the JcpConnection object has just been created.

Parameters:

`connectionevent` - JcpConnectionEvent.

---

connectionAlerting

`public void` **`connectionAlerting`**`(JcpConnectionEvent connectionevent)`

Indicates that the JcpConnection has just been placed in the JcpConnection.ALERTING state

Parameters:

`connectionevent` - JcpConnectionEvent.

---

connectionConnected

`public void` **`connectionConnected`**`(JcpConnectionEvent connectionevent)`

Indicates that the JcpConnection has just been placed in the JcpConnection.CONNECTED state

Parameters:

`connectionevent` - JcpConnectionEvent.

---

connectionInProgress

`public void` **`connectionInProgress`**`(JcpConnectionEvent connectionevent)`

Indicates that the JcpConnection has just been placed in the JcpConnection.INPROGRESS state

Parameters:

`connectionevent` - JcpConnectionEvent.

---

connectionFailed

`public void` **`connectionFailed`**`(JcpConnectionEvent connectionevent)`

Indicates that the JcpConnection has just been placed in the JcpConnection.FAILED state

Parameters:

`connectionevent` - JcpConnectionEvent.

---

connectionDisconnected

`public void` **`connectionDisconnected`**`(JcpConnectionEvent connectionevent)`

Indicates that the JcpConnection has just been placed in the JcpConnection.DISCONNECTED state

Parameters:

`connectionevent` - JcpConnectionEvent.

---

connectionUnknown

`public void` **`connectionUnknown`**`(JcpConnectionEvent connectionevent)`

Indicates that the Connection has just been placed in the JcpConnection.UNKNOWN state

Parameters:

`connectionevent` - JcpConnectionEvent.

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcp.JcpConnectionListener

Packages that use JcpConnectionListener

jain.jcc

Uses of JcpConnectionListener in jain.jcc

Subinterfaces of JcpConnectionListener in jain.jcc	
interface	**JccConnectionListener** This interface is an extension of the JccCallListener and the JcpConnectionListener interface and reports state changes both of the JccCall and its JccConnections.

Methods in jain.jcc with parameters of type JcpConnectionListener	
void	**JccProvider.addConnectionListener**(JcpConnectionListener connectionlistener, EventFilter filter) Add a connection listener to all connections under this JcpProvider.
void	**JccProvider.removeConnectionListener**(JcpConnectionListener connectionlistener) Removes a connection listener that was registered using this.addConnectionListener.
void	**JccCall.addConnectionListener**(JcpConnectionListener cl, EventFilter filter) Add a connection listener to all connections under this call.
void	**JccCall.removeConnectionListener**(JcpConnectionListener cl) Removes the connection listener from all connections under this call.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Interface JcpConnectionEvent

All Known Subinterfaces:

JccConnectionEvent

public interface **JcpConnectionEvent**
extends JcpCallEvent

This is the base interface for all JcpConnection related events. This interface extends the JcpCallEvent interface and therefore is reported via the JcpCallListener interface.

Field Summary

static int	**CONNECTION_ALERTING** This event indicates that the state of the JcpConnection object has changed to JcpConnection.ALERTING.
static int	**CONNECTION_CONNECTED** This event indicates that the state of the JcpConnection object has changed to JcpConnection.CONNECTED.
static int	**CONNECTION_CREATED** This event indicates that a new JcpConnection object has been created in the JcpConnection.IDLE state.
static int	**CONNECTION_DISCONNECTED** This event indicates that the state of the JcpConnection object has changed to JcpConnection.DISCONNECTED.
static int	**CONNECTION_FAILED** This event indicates that the state of the JcpConnection object has changed to JcpConnection.FAILED.
static int	**CONNECTION_INPROGRESS** This event indicates that the state of the JcpConnection object has changed to JcpConnection.INPROGRESS.
static int	**CONNECTION_UNKNOWN** This event indicates that the state of the JcpConnection object has changed to JcpConnection.UNKNOWN.

Fields inherited from interface jain.jcp.JcpCallEvent
CALL_ACTIVE, CALL_CREATED, CALL_EVENT_TRANSMISSION_ENDED, CALL_INVALID

Fields inherited from interface jain.jcp.Event
CAUSE_CALL_CANCELLED, CAUSE_DEST_NOT_OBTAINABLE, CAUSE_INCOMPATIBLE_DESTINATION, CAUSE_LOCKOUT, CAUSE_NETWORK_CONGESTION, CAUSE_NETWORK_NOT_OBTAINABLE, CAUSE_NEW_CALL, CAUSE_NORMAL, CAUSE_REDIRECTED, CAUSE_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN

Method Summary

JcpConnection	**getConnection**() Returns the JcpConnection associated with this event.

Methods inherited from interface jain.jcp.JcpCallEvent
getCall

Methods inherited from interface jain.jcp.Event
getCause, getID, getSource

Field Detail

CONNECTION_ALERTING

```
public static final int CONNECTION_ALERTING
```

This event indicates that the state of the JcpConnection object has changed to JcpConnection.ALERTING.

CONNECTION_CONNECTED

`public static final int` **`CONNECTION_CONNECTED`**

This event indicates that the state of the JcpConnection object has changed to JcpConnection.CONNECTED.

CONNECTION_CREATED

`public static final int` **`CONNECTION_CREATED`**

This event indicates that a new JcpConnection object has been created in the JcpConnection.IDLE state.

CONNECTION_DISCONNECTED

`public static final int` **`CONNECTION_DISCONNECTED`**

This event indicates that the state of the JcpConnection object has changed to JcpConnection.DISCONNECTED.

CONNECTION_FAILED

`public static final int` **`CONNECTION_FAILED`**

This event indicates that the state of the JcpConnection object has changed to JcpConnection.FAILED.

CONNECTION_INPROGRESS

`public static final int` **`CONNECTION_INPROGRESS`**

This event indicates that the state of the JcpConnection object has changed to JcpConnection.INPROGRESS.

CONNECTION_UNKNOWN

`public static final int` **`CONNECTION_UNKNOWN`**

This event indicates that the state of the JcpConnection object has changed to

JcpConnection.UNKNOWN.

Method Detail

getConnection

`public JcpConnection getConnection()`

Returns the JcpConnection associated with this event.

Returns:

JcpConnection associated with this JcpConnection Event.

Overview Package Class Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS **FRAMES NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcp.JcpConnectionEvent

Packages that use JcpConnectionEvent

jain.jcc	
jain.jcp	

Uses of JcpConnectionEvent in jain.jcc

Subinterfaces of JcpConnectionEvent in jain.jcc	
interface	**JccConnectionEvent** This is the base interface for all JccConnection related events.

Uses of JcpConnectionEvent in jain.jcp

Methods in jain.jcp with parameters of type JcpConnectionEvent	
void	**JcpConnectionListener.connectionCreated**(JcpConnectionEvent connectionevent) Indicates that the JcpConnection object has just been created.
void	**JcpConnectionListener.connectionAlerting**(JcpConnectionEvent connectionevent) Indicates that the JcpConnection has just been placed in the JcpConnection.ALERTING state
void	**JcpConnectionListener.connectionConnected**(JcpConnectionEvent connectionevent) Indicates that the JcpConnection has just been placed in the JcpConnection.CONNECTED state
void	**JcpConnectionListener.connectionInProgress**(JcpConnectionEvent connectionevent) Indicates that the JcpConnection has just been placed in the JcpConnection.INPROGRESS state
void	**JcpConnectionListener.connectionFailed**(JcpConnectionEvent connectionevent) Indicates that the JcpConnection has just been placed in the JcpConnection.FAILED state
void	**JcpConnectionListener.connectionDisconnected**(JcpConnectionEvent connectionevent) Indicates that the JcpConnection has just been placed in the JcpConnection.DISCONNECTED state
void	**JcpConnectionListener.connectionUnknown**(JcpConnectionEvent connectionevent) Indicates that the Connection has just been placed in the JcpConnection.UNKNOWN state

jain.jcp

Interface JcpCallEvent

All Known Subinterfaces:

JccCallEvent, JccConnectionEvent, JcpConnectionEvent

public interface **JcpCallEvent**
extends Event

This is the base interface for all JcpCall-related events. Events which extend this interface are reported via the JcpCallListener method.

Field Summary

static int	**CALL_ACTIVE** The CALL_ACTIVE event indicates that the state of the Call object has changed to JcpCall.ACTIVE.
static int	**CALL_CREATED** The CALL_CREATED event indicates that the JcpCall object has been created and is in the IDLE state.
static int	**CALL_EVENT_TRANSMISSION_ENDED** The CALL_EVENT_TRANSMISSION_ENDED event indicates that the application will no longer receive JcpCall events on the instance of the JcpCallListener.
static int	**CALL_INVALID** The CALL_INVALID event indicates that the state of the JcpCall object has changed to JcpCall.INVALID.

Fields inherited from interface jain.jcp.Event

CAUSE_CALL_CANCELLED, CAUSE_DEST_NOT_OBTAINABLE, CAUSE_INCOMPATIBLE_DESTINATION, CAUSE_LOCKOUT, CAUSE_NETWORK_CONGESTION, CAUSE_NETWORK_NOT_OBTAINABLE, CAUSE_NEW_CALL, CAUSE_NORMAL, CAUSE_REDIRECTED, CAUSE_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN

Method Summary

JcpCall	**getCall**() Returns the JcpCall object associated with this event.

Methods inherited from interface jain.jcp.Event
getCause, getID, getSource

Field Detail

CALL_ACTIVE

```
public static final int CALL_ACTIVE
```

The CALL_ACTIVE event indicates that the state of the Call object has changed to JcpCall.ACTIVE.
This constant corresponds to a specific call state change, is passed via a JcpCallEvent event and is reported to the JcpCallListener.callActive method.

CALL_INVALID

```
public static final int CALL_INVALID
```

The CALL_INVALID event indicates that the state of the JcpCall object has changed to JcpCall.INVALID.
This constant corresponds to a specific call state change, is passed via a JcpCallEvent event and is reported to the JcpCallListener.callInvalid method.

CALL_EVENT_TRANSMISSION_ENDED

```
public static final int CALL_EVENT_TRANSMISSION_ENDED
```

The CALL_EVENT_TRANSMISSION_ENDED event indicates that the application will no longer receive JcpCall events on the instance of the JcpCallListener.
This constant corresponds to a specific call state change, is passed via a JcpCallEvent event and is reported to the JcpCallListener.callEventTransmissionEnded method.

CALL_CREATED

`public static final int` **`CALL_CREATED`**

The CALL_CREATED event indicates that the JcpCall object has been created and is in the IDLE state.
This constant corresponds to a specific call state change, is passed via a JcpCallEvent event and is reported to the JcpCallListener.callcreated method.

Method Detail

getCall

`public` `JcpCall` **`getCall`**`()`

Returns the JcpCall object associated with this event.

Returns:

JcpCall represents the JcpCall object associated with this JcpCall event.

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Uses of Interface jain.jcp.JcpCallEvent

Packages that use JcpCallEvent

jain.jcc	
jain.jcp	

Uses of JcpCallEvent in jain.jcc

Subinterfaces of JcpCallEvent in jain.jcc	
interface	**JccCallEvent** This is the base interface for all JccCall-related events.
interface	**JccConnectionEvent** This is the base interface for all JccConnection related events.

Uses of JcpCallEvent in jain.jcp

Subinterfaces of JcpCallEvent in jain.jcp	
interface	**JcpConnectionEvent** This is the base interface for all JcpConnection related events.

Methods in jain.jcp with parameters of type JcpCallEvent	
void	**JcpCallListener.callActive**(JcpCallEvent callevent) Indicates that the state of the JcpCall object has changed to JcpCall.ACTIVE.
void	**JcpCallListener.callInvalid**(JcpCallEvent callevent) Indicates that the state of the JcpCall object has changed to JcpCall.INVALID.
void	**JcpCallListener.callEventTransmissionEnded**(JcpCallEvent callevent) This method is called to indicate that the application will no longer receive JcpCallEvent events on the instance of the JcpCallListener.

void	**JcpCallListener.callcreated**(JcpCallEvent callevent) Indicates that the state of the JcpCall object has changed to JcpCall.IDLE.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Interface JcpPeer

public interface **JcpPeer**

The JcpPeer interface represents a vendor's particular implementation of the JCP API.
Since implementations of just the JCP interface are not expected, other interfaces derived from JCP for example, JTAPI, JCC etc are expected to implement this interface. The JcpPeer object returned by the JcpPeerFactory.getJcpPeer() method determines which JcpProviders are made available to the application.
Applications use the JcpPeer.getProvider() method on this interface to obtain new JcpProvider objects. Each implementation may support one or more different "services". A list of available services can be obtained via the JcpPeer.getServices() method.

Obtaining a JcpProvider

Applications use the `JcpPeer.getProvider()` method on this interface to obtain new JcpProvider objects. Each implementation may support one or more different "services" (e.g. for different types of underlying network substrate). A list of available services can be obtained via the `JcpPeer.getServices()` method.

Applications may also supply optional arguments to the JcpProvider through the `JcpPeer.getProvider()` method. These arguments are appended to the `providerString` argument passed to the `JcpPeer.getProvider()` method. The `providerString` argument has the following format:

< service name > ; arg1 = val1; arg2 = val2; ...

Where < service name > is not optional, and each optional argument pair which follows is separated by a semi-colon. The keys for these arguments is implementation specific, except for two standard-defined keys:

1. login: provides the login user name to the Provider.
2. passwd: provides a password to the Provider.

Method Summary

`java.lang.String`	**`getName()`** Returns the name of this JcpPeer object instance.

jain.jcp.Provider	**getProvider**(java.lang.String providerString) Returns an instance of a Provider object given a string argument which contains the desired service name.
java.lang.String[]	**getServices**() Returns the services that this implementation supports.

Method Detail

getName

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

Returns the name of this JcpPeer object instance. The name is the same name used as an argument to JcpPeerFactory.getJcpPeer() method.

Returns:

The name of this JcpPeer object instance.

---

getServices

public java.lang.String[] **getServices**()

Returns the services that this implementation supports. This method returns null if no services are supported.

Returns:

services that this implementation supports.

---

getProvider

public jain.jcp.Provider **getProvider**(java.lang.String providerString)
throws ProviderUnavailableException

Returns an instance of a Provider object given a string argument which contains the desired service name. Optional arguments may also be provided in this string, with the following format:

< service name > ; arg1 = val1; arg2 = val2; ...

Where < service name > is not optional, and each optional argument pair which follows is separated by a semi-colon. The keys for these arguments is implementation specific, except

for two standard-defined keys:

1. login: provides the login user name to the Provider.
2. passwd: provides a password to the Provider.

If the argument is null, this method returns some default Provider as determined by the JCPPeer object. The returned Provider is not in the `Provider.SHUTDOWN` state.Note that this may also result in the application obtaining a reference to a Provider which has already been created.

Post-conditions:

1. this.getProvider().getState() != JcpProvider.SHUTDOWN

Parameters:

`providerString` - is the name of the desired service.

Returns:

An instance of the Provider object.

Throws:

ProviderUnavailableException - indicates a Provider corresponding to the given string is unavailable.

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcp.JcpPeer

Packages that use JcpPeer

jain.jcp

Uses of JcpPeer in jain.jcp

Methods in jain.jcp that return JcpPeer	
`static JcpPeer`	**`JcpPeerFactory.getJcpPeer`**`(java.lang.String jcpPeerName)` Returns an instance of a JcpPeer object given a fully qualified classname of the class which implements the JcpPeer object.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Class JcpPeerFactory

```
java.lang.Object
  |
  +--jain.jcp.JcpPeerFactory
```

public class **JcpPeerFactory**
extends java.lang.Object

The `JcpPeerFactory` class is a class by which applications obtain a JcpProvider object.

Introduction

Applications use this class to first obtain a class which implements the `JcpPeer` interface. The `JcpPeer` interface represents a particular vendor's implementation of JCP. The term 'peer' is Java nomenclature for "a particular platform-specific implementation of a Java interface or API". This term has the same meaning for the Java Core Package API. Applications are not permitted to create an instance of the `JcpPeerFactory` class. Through an installation procedure provided by each implementator, a `JcpPeer` class is made available to an application environment. When applications have a JcpPeer object for a particular platform-dependent implementation, they may obtain a JcpProvider object via that interface. The details of that interface are discussed in the specification for the `JcpPeer` interface.

Obtaining a JcpPeer Object

Applications use the `JcpPeerFactory.getJcpPeer()` method to obtain a JcpPeer object. The argument to this method is a classname which represents an object which implements the `JcpPeer` interface. This object and the classname under which it can be found must be supplied by the vendor of the implementation. Note that this object is not a JcpProvider, however, this interface is used to obtain JcpProvider objects from that particular implementation.

The JCP places conventions on vendors on the classname they use for their JcpPeer object. This class name *must* begin with the domain name assigned to the vendor in reverse order. Because the space of domain names is managed, this scheme ensures that collisions between two different vendor's implementations will not happen. For example, an implementation from Sun Microsystem's will have "com.sun" as the prefix to its JcpPeer class. After the reversed domain name, vendors are free to choose any class hierarchy they desire.

Default JcpPeer

Additionally, the vendor providing the JcpPeer class may supply a a `DefaultJcpPeer.class` class file. When placed in the classpath of applications, this class (which must implement the `JcpPeer` interface) becomes the default JcpPeer object returned by the `JcpPeerFactory.getJcpPeer()` method. By convention the default class name must be `DefaultJcpPeer`.

In basic environments, applications and users do not want the burden of finding out the class name in order to use a particular implementation. Therefore, the `JcpPeerFactory` class supports a mechanism for applications to obtain the default implementation for their system. If applications use a `null` argument to the `JcpPeerFactory.getJcpPeer()` method, they will be returned the default installed implementation on their system if it exists.

Note: It is the responsibility of implementation vendors to supply a version of a `DefaultJcpPeer` or some means to alias their peer implementation along with a means to place that `DefaultJcpPeer` class in the application classpath.

Method Summary

`static JcpPeer`	**`getJcpPeer`**`(java.lang.String jcpPeerName)` Returns an instance of a JcpPeer object given a fully qualified classname of the class which implements the JcpPeer object.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Method Detail

getJcpPeer

```
public static JcpPeer getJcpPeer(java.lang.String jcpPeerName)
                          throws JcpPeerUnavailableException
```

Returns an instance of a JcpPeer object given a fully qualified classname of the class which implements the JcpPeer object.

If no classname is provided (null), a default class named `DefaultJcpPeer` is chosen as the classname to load. If it does not exist or is not installed in the CLASSPATH as the default, a JcpPeerUnavailableException exception is thrown.

Parameters:

`jcpPeerName` - The classname of the JcpPeer object class.

Returns:

An instance of the JcpPeer object.

Throws:

JcpPeerUnavailableException - Indicates that the JcpPeer specified by the classname is not available.

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Class jain.jcp.JcpPeerFactory

No usage of jain.jcp.JcpPeerFactory

Overview Package Class Use Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Class JcpPeerUnavailableException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--jain.jcp.JcpPeerUnavailableException
```

public class **JcpPeerUnavailableException**

extends java.lang.Exception

This Exception indicates that the JcpPeer is unavailable on the current system.

See Also:

Serialized Form

Constructor Summary

JcpPeerUnavailableException() Constructor with no string.
JcpPeerUnavailableException(java.lang.String s) Constructor with string.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

JcpPeerUnavailableException

`public` **`JcpPeerUnavailableException`**`()`

Constructor with no string.

JcpPeerUnavailableException

`public` **`JcpPeerUnavailableException`**`(java.lang.String s)`

Constructor with string.

Parameters:

`s` - description of the fault.

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Class jain.jcp.JcpPeerUnavailableException

Packages that use JcpPeerUnavailableException

jain.jcp

Uses of JcpPeerUnavailableException in jain.jcp

Methods in jain.jcp that throw JcpPeerUnavailableException	
`static JcpPeer`	**`JcpPeerFactory.getJcpPeer`**`(java.lang.String jcpPeerName)` Returns an instance of a JcpPeer object given a fully qualified classname of the class which implements the JcpPeer object.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Class InvalidStateException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--jain.jcp.InvalidStateException
```

public class **InvalidStateException**

extends java.lang.Exception

An InvalidStateException indicates that that current state of an object involved in the method invocation does not meet the acceptable pre-conditions for the method. Each method which changes the call model typically has a set of states in which the object must be as a pre-condition for the method. Each method documents the pre-condition states for objects. Typically, this method might succeed later when the object in question reaches the proper state.

This exception provides the application with the object in question and the state it is currently in.

See Also:

Serialized Form

Field Summary

static int	**ADDRESS_OBJECT** The invalid object in question is the Address
static int	**CALL_OBJECT** The invalid object in question is the Call
static int	**CONNECTION_OBJECT** The invalid object in question is the Connection
static int	**PROVIDER_OBJECT** The invalid object in question is the Provider

Constructor Summary

InvalidStateException(java.lang.Object object, int type, int state) Constructor with no string.
InvalidStateException(java.lang.Object object, int type, int state, java.lang.String s) Constructor which takes a string description.

Method Summary

java.lang.Object	**getObject**() Returns the object which has the incorrect state.
int	**getObjectType**() Returns the type of object in question.
int	**getState**() Returns the state of the object.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

PROVIDER_OBJECT

public static final int **PROVIDER_OBJECT**

The invalid object in question is the Provider

CALL_OBJECT

```
public static final int CALL_OBJECT
```

The invalid object in question is the Call

CONNECTION_OBJECT

```
public static final int CONNECTION_OBJECT
```

The invalid object in question is the Connection

ADDRESS_OBJECT

```
public static final int ADDRESS_OBJECT
```

The invalid object in question is the Address

Constructor Detail

InvalidStateException

```
public InvalidStateException(java.lang.Object object,
                             int type,
                             int state)
```

Constructor with no string.

Parameters:

`object` - instance associated with the invalid sate.

`type` - type of failure

`state` - current state at time of fault

InvalidStateException

```
public InvalidStateException(java.lang.Object object,
                             int type,
                             int state,
                             java.lang.String s)
```

Constructor which takes a string description.

Parameters:

`object` - instance associated with the invalid sate.

`type` - type of failure

`state` - current state at time of fault

`s` - desciption of the fault

Method Detail

getObjectType

`public int` **`getObjectType`**`()`

Returns the type of object in question.

Returns:

The type of object in question.

---

getObject

`public java.lang.Object` **`getObject`**`()`

Returns the object which has the incorrect state.

Returns:

The object which is in the wrong state.

---

getState

`public int` **`getState`**`()`

Returns the state of the object.

Returns:

The state of the object.

Uses of Class jain.jcp.InvalidStateException

Packages that use InvalidStateException

jain.jcc	
jain.jcp	

Uses of InvalidStateException in jain.jcc

Methods in jain.jcc that throw InvalidStateException	
void	**JccConnection.routeConnection**(boolean attachmedia) Routes this JccConnection to the specified target address.
void	**JccConnection.release**() Drops a JccConnection from an active telephone call.
void	**JccCall.addConnectionListener**(JcpConnectionListener cl, EventFilter filter) Add a connection listener to all connections under this call.
void	**JccCall.release**() This method requests the release of the call object and associated connection objects.
JcpConnection	**JccCall.createConnection**(java.lang.String targetAddress, java.lang.String originatingAddress, java.lang.String originalCalledAddress, java.lang.String redirectingAddress) Creates a new JccConnection and attaches it to this JccCall.
JcpConnection	**JccCall.routeCall**(java.lang.String targetAddress, java.lang.String originatingAddress, java.lang.String originalDestinationAddress, java.lang.String redirectingAddress) This method requests routing of a call to the given call party.

Uses of InvalidStateException in jain.jcp

Methods in jain.jcp that throw InvalidStateException	
JcpCall	**JcpProvider.createCall()** Creates a new instance of the call with no connections.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Class InvalidPartyException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--jain.jcp.InvalidPartyException
```

public class **InvalidPartyException**

extends java.lang.Exception

This exception indicates that a party given as an argument to the method call was invalid. This may either be the originating party of a telephone call or the destination party of a telephone call.

See Also:

Serialized Form

Field Summary

static int	**DESTINATION_PARTY** Indicates that the destination party was invalid.
static int	**ORIGINATING_PARTY** Indicates that the originating party was invalid.
static int	**UNKNOWN_PARTY** Indicates that the party was unknown.

Constructor Summary

InvalidPartyException(int type) Constructor with no string.
InvalidPartyException(int type, java.lang.String s) Constructor which takes a string description.

Method Summary

int	**getType**() Returns the type of party.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

ORIGINATING_PARTY

```
public static final int ORIGINATING_PARTY
```

Indicates that the originating party was invalid.

DESTINATION_PARTY

```
public static final int DESTINATION_PARTY
```

Indicates that the destination party was invalid.

UNKNOWN_PARTY

```
public static final int UNKNOWN_PARTY
```

Indicates that the party was unknown.

Constructor Detail

InvalidPartyException

```
public InvalidPartyException(int type)
```

Constructor with no string.

Parameters:

`type` - the type of party expected.

InvalidPartyException

```
public InvalidPartyException(int type,
                             java.lang.String s)
```

Constructor which takes a string description.

Parameters:

`type` - type of exception

`s` - description of the fault

Method Detail

getType

```
public int getType()
```

Returns the type of party.

Returns:

The type of party.

Uses of Class jain.jcp.InvalidPartyException

Packages that use InvalidPartyException

jain.jcc

Uses of InvalidPartyException in jain.jcc

Methods in jain.jcc that throw InvalidPartyException	
void	**JccConnection.routeConnection**(boolean attachmedia) Routes this JccConnection to the specified target address.
JcpConnection	**JccCall.routeCall**(java.lang.String targetAddress, java.lang.String originatingAddress, java.lang.String originalDestinationAddress, java.lang.String redirectingAddress) This method requests routing of a call to the given call party.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Class InvalidArgumentException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--jain.jcp.InvalidArgumentException
```

public class **InvalidArgumentException**
extends java.lang.Exception

This Exception indicates that an invalid argument is passed into a method.

See Also:

Serialized Form

Constructor Summary

InvalidArgumentException() Constructor with no String.
InvalidArgumentException(java.lang.String s) Constructor which takes a string description.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

InvalidArgumentException

`public` **`InvalidArgumentException`**`()`

Constructor with no String.

InvalidArgumentException

`public` **`InvalidArgumentException`**`(java.lang.String s)`

Constructor which takes a string description.

Parameters:

`s` - description of the faulty argument.

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Uses of Class jain.jcp.InvalidArgumentException

Packages that use InvalidArgumentException

jain.jcc	
jain.jcp	

Uses of InvalidArgumentException in jain.jcc

Methods in jain.jcc that throw InvalidArgumentException	
void	**JccConnection.routeConnection**(boolean attachmedia) Routes this JccConnection to the specified target address.
JcpConnection	**JccCall.routeCall**(java.lang.String targetAddress, java.lang.String originatingAddress, java.lang.String originalDestinationAddress, java.lang.String redirectingAddress) This method requests routing of a call to the given call party.

Uses of InvalidArgumentException in jain.jcp

Methods in jain.jcp that throw InvalidArgumentException	
JcpAddress	**JcpProvider.getAddress**(java.lang.String address) Returns an Address object which corresponds to the (telephone) number string provided.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Serialized Form

Package jain.jcp

Class jain.jcp.InvalidArgumentException implements Serializable

Class jain.jcp.InvalidPartyException implements Serializable

Serialized Fields

_type

`int` **`_type`**

This private variable stores the type of party.

Class jain.jcp.InvalidStateException implements Serializable

Serialized Fields

_object

`java.lang.Object` **`_object`**

The current object.

_state

`int` **`_state`**

The current state.

_type

`int` **`_type`**

The type of the party.

Class jain.jcp.JcpPeerUnavailableException implements Serializable

Class jain.jcp.MethodNotSupportedException implements Serializable

Class jain.jcp.PlatformException implements Serializable

Class jain.jcp.PrivilegeViolationException implements Serializable

Serialized Fields

_type

`int` **`_type`**

This private variable stores the type of privilege not available.

Class jain.jcp.ProviderUnavailableException implements Serializable

Serialized Fields

_cause

`int` **`_cause`**

This private variable holds the cause for this exception.

Class jain.jcp.ResourceUnavailableException implements Serializable

Serialized Fields

_type

`int` **`_type`**

This private variable stores the type of resource.

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Class MethodNotSupportedException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--jain.jcp.MethodNotSupportedException
```

public class **MethodNotSupportedException**
extends java.lang.Exception

This Exception indicates that the method which was invoked is not supported by the implementation.

See Also:

Serialized Form

Constructor Summary

MethodNotSupportedException() Constructor with no string.
MethodNotSupportedException(java.lang.String s) Constructor with a string description.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

MethodNotSupportedException

`public` **`MethodNotSupportedException`**`()`

Constructor with no string.

MethodNotSupportedException

`public` **`MethodNotSupportedException`**`(java.lang.String s)`

Constructor with a string description.

Parameters:

`s` - description of the fault.

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Uses of Class jain.jcp.MethodNotSupportedException

Packages that use MethodNotSupportedException

jain.jcc	
jain.jcp	

Uses of MethodNotSupportedException in jain.jcc

Methods in jain.jcc that throw MethodNotSupportedException

void	**JccProvider.addProviderListener**(JcpProviderListener providerlistener, EventFilter filter) Adds a listener to this provider.
void	**JccProvider.addCallListener**(JcpCallListener calllistener) Add a call listener to all call objects that will be created under the domain of this provider.
void	**JccProvider.addCallListener**(JcpCallListener calllistener, EventFilter filter) Add a call listener to all call objects that will be created under the domain of this provider.
void	**JccProvider.setCallLoadControl**(JcpAddress[] address, double duration, double[] mechanism, int[] treatment) This method imposes or removes load control on calls made to the specified addresses.
void	**JccProvider.addCallLoadControlListener**(CallLoadControlListener loadcontrollistener, EventFilter filter) Adds a listener to listen to load control related events.
void	**JccConnection.routeConnection**(boolean attachmedia) Routes this JccConnection to the specified target address.
void	**JccCall.addCallListener**(JcpCallListener calllistener, EventFilter filter) Add a listener to this call.
JcpConnection	**JccCall.createConnection**(java.lang.String targetAddress, java.lang.String originatingAddress, java.lang.String originalCalledAddress, java.lang.String redirectingAddress) Creates a new JccConnection and attaches it to this JccCall.
JcpConnection	**JccCall.routeCall**(java.lang.String targetAddress, java.lang.String originatingAddress, java.lang.String originalDestinationAddress, java.lang.String redirectingAddress) This method requests routing of a call to the given call party.
void	**JccCall.superviseCall**(JccCallListener calllistener, double time, int treatment, double bytes) The application calls this method to supervise a call.

Uses of MethodNotSupportedException in jain.jcp

Methods in jain.jcp that throw MethodNotSupportedException

JcpCall	**JcpProvider.createCall()** Creates a new instance of the call with no connections.
void	**JcpProvider.addProviderListener**(JcpProviderListener providerlistener) Adds a listener to this provider.
void	**JcpCall.addCallListener**(JcpCallListener calllistener) Add a listener to this call.

Overview Package Class Use Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcc

Interface CallLoadControlListener

public interface **CallLoadControlListener**

extends java.util.EventListener

Interface for notifying load control related changes happening in a JccProvider event. These changes are reported as events to the CallLoadControlListener method corresponding to the type of event. Applications must instantiate an object which implements this interface and then use the JccProvider.addCallLoadControlListener() method to register the object to receive all future events associated with the JccProvider object.

Method Summary

void	**providerCallOverloadCeased**(CallLoadControlEvent loadcontrolevent) This method indicates that the network has detected that the overload has ceased and has automatically removed load control on calls requested to a particular address range or calls made to a particular destination.
void	**providerCallOverloadEncountered**(CallLoadControlEvent loadcontrolevent) This method indicates that the network has detected overload and may have automatically imposed load control on calls requested to a particular address range or calls made to a particular destination.

Method Detail

providerCallOverloadEncountered

public void **providerCallOverloadEncountered**(CallLoadControlEvent loadcontrolevent)

This method indicates that the network has detected overload and may have automatically imposed load control on calls requested to a particular address range or calls made to a particular destination.

Parameters:

loadcontrolevent - CallLoadControlEvent with event ID CallLoadControlEvent.PROVIDER_CALL_OVERLOAD_ENCOUNTERED.

providerCallOverloadCeased

public void **providerCallOverloadCeased**(CallLoadControlEvent loadcontrolevent)

This method indicates that the network has detected that the overload has ceased and has automatically removed load control on calls requested to a particular address range or calls made to a particular destination.

Parameters:

`loadcontrolevent` - CallLoadControlEvent with event ID CallLoadControlEvent.PROVIDER_CALL_OVERLOAD_CEASED.

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcc.CallLoadControlListener

Packages that use CallLoadControlListener

jain.jcc

Uses of CallLoadControlListener in jain.jcc

Methods in jain.jcc with parameters of type CallLoadControlListener	
void	**JccProvider.addCallLoadControlListener**(CallLoadControlListener loadcontrollistener, EventFilter filter) Adds a listener to listen to load control related events.
void	**JccProvider.removeCallLoadControlListener**(CallLoadControlListener loadcontrollistener) Deregisters the load control listener.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Class PlatformException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.lang.RuntimeException
                    |
                    +--jain.jcp.PlatformException
```

public class **PlatformException**

extends java.lang.RuntimeException

A PlatformException indicates an implementation specific exception. The specific exceptions which implementations throw would be documented in their release notes.

See Also:

Serialized Form

Constructor Summary

PlatformException() Constructor with no string.
PlatformException(java.lang.String s) Constructor which takes a string description.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

PlatformException

public **PlatformException**()

Constructor with no string.

PlatformException

public **PlatformException**(java.lang.String s)

Constructor which takes a string description.

Parameters:

s - description of the fault.

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Uses of Class jain.jcp.PlatformException

No usage of jain.jcp.PlatformException

Overview Package Class Use Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Class PrivilegeViolationException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--jain.jcp.PrivilegeViolationException
```

public class **PrivilegeViolationException**

extends java.lang.Exception

This exception indicates that an action pertaining to a certain object failed because the application did not have the proper security permissions to execute that command.
This class stores the type of privilege not available which is obtained via the getType( ) method in this class.

See Also:

Serialized Form

Field Summary

static int	**DESTINATION_VIOLATION** A privilege violation occurred at the destination.
static int	**ORIGINATOR_VIOLATION** A privilege violation occurred at the origination.
static int	**UNKNOWN_VIOLATION** A privilege violation occurred at an unknown place.

Constructor Summary

PrivilegeViolationException(int type) Constructor takes no string.

PrivilegeViolationException(int type, java.lang.String s) Constructor takes a string.

Method Summary

int	**getType**() Returns the type of privilege which is not available.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

ORIGINATOR_VIOLATION

public static final int **ORIGINATOR_VIOLATION**

A privilege violation occurred at the origination.

DESTINATION_VIOLATION

public static final int **DESTINATION_VIOLATION**

A privilege violation occurred at the destination.

UNKNOWN_VIOLATION

public static final int **UNKNOWN_VIOLATION**

A privilege violation occurred at an unknown place.

Constructor Detail

PrivilegeViolationException

```
public PrivilegeViolationException(int type)
```

Constructor takes no string.

Parameters:

`type` - kind of violation.

PrivilegeViolationException

```
public PrivilegeViolationException(int type,
                                   java.lang.String s)
```

Constructor takes a string.

Parameters:

`type` - kind of violation.

`s` - description of the violation.

Method Detail

getType

```
public int getType()
```

Returns the type of privilege which is not available.

Returns:

The type of privilege.

Uses of Class jain.jcp.PrivilegeViolationException

Packages that use PrivilegeViolationException

jain.jcc	
jain.jcp	

Uses of PrivilegeViolationException in jain.jcc

Methods in jain.jcc that throw PrivilegeViolationException	
void	**JccConnection.routeConnection**(boolean attachmedia) Routes this JccConnection to the specified target address.
void	**JccConnection.release**() Drops a JccConnection from an active telephone call.
void	**JccCall.release**() This method requests the release of the call object and associated connection objects.
JcpConnection	**JccCall.createConnection**(java.lang.String targetAddress, java.lang.String originatingAddress, java.lang.String originalCalledAddress, java.lang.String redirectingAddress) Creates a new JccConnection and attaches it to this JccCall.
JcpConnection	**JccCall.routeCall**(java.lang.String targetAddress, java.lang.String originatingAddress, java.lang.String originalDestinationAddress, java.lang.String redirectingAddress) This method requests routing of a call to the given call party.

Uses of PrivilegeViolationException in jain.jcp

Methods in jain.jcp that throw PrivilegeViolationException	
JcpCall	**JcpProvider.createCall()** Creates a new instance of the call with no connections.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Class ProviderUnavailableException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.lang.RuntimeException
                    |
                    +--jain.jcp.ProviderUnavailableException
```

public class **ProviderUnavailableException**

extends java.lang.RuntimeException

This exception indicates that the Provider is currently not available to the application. This exception is typically thrown in two cases: when JcpPeer.getProvider() is called or on any method when the Provider is in a SHUTDOWN state.

The exception stores the reason for the failure which may be obtained via the getCause() method on this interface.

See Also:

Serialized Form

Field Summary

static int	**CAUSE_INVALID_ARGUMENT** Constant definition for an invalid optional argument given to JtapiPeer.getProvider().
static int	**CAUSE_INVALID_SERVICE** Constant definition for an invalid service string given to JtapiPeer.getProvider().
static int	**CAUSE_NOT_IN_SERVICE** Constant definition for the Provider not in the "in service" state.

static int	**CAUSE_UNKNOWN** Constant definition for an unknown cause.

Constructor Summary

ProviderUnavailableException() Constructor with no cause and string.
ProviderUnavailableException(int cause) Constructor which takes a cause string.
ProviderUnavailableException(int cause, java.lang.String s) Constructor which takes both a string and a cause.
ProviderUnavailableException(java.lang.String s) Constructor which takes a string description.

Method Summary

int	**getCause**() Returns the cause for this exception.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

CAUSE_UNKNOWN

public static final int **CAUSE_UNKNOWN**

Constant definition for an unknown cause.

CAUSE_NOT_IN_SERVICE

`public static final int` **`CAUSE_NOT_IN_SERVICE`**

Constant definition for the Provider not in the "in service" state.

CAUSE_INVALID_SERVICE

`public static final int` **`CAUSE_INVALID_SERVICE`**

Constant definition for an invalid service string given to `JtapiPeer.getProvider()`.

CAUSE_INVALID_ARGUMENT

`public static final int` **`CAUSE_INVALID_ARGUMENT`**

Constant definition for an invalid optional argument given to `JtapiPeer.getProvider()`.

Constructor Detail

ProviderUnavailableException

`public` **`ProviderUnavailableException`**`()`

Constructor with no cause and string.

ProviderUnavailableException

`public` **`ProviderUnavailableException`**`(int cause)`

Constructor which takes a cause string.

Parameters:

`cause` - reason code for this fault

ProviderUnavailableException

`public` **`ProviderUnavailableException`**`(java.lang.String s)`

Constructor which takes a string description.

Parameters:

`s` - description of the fault

ProviderUnavailableException

```
public ProviderUnavailableException(int cause,
                                    java.lang.String s)
```

Constructor which takes both a string and a cause.

Parameters:

`cause` - reason code for the fault

`s` - description of the fault

Method Detail

getCause

```
public int getCause()
```

Returns the cause for this exception.

Returns:

The cause of this exception.

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Class jain.jcp.ProviderUnavailableException

Packages that use ProviderUnavailableException

jain.jcp	

Uses of ProviderUnavailableException in jain.jcp

Methods in jain.jcp that throw ProviderUnavailableException	
`jain.jcp.Provider`	**`JcpPeer.getProvider`**`(java.lang.String providerString)` Returns an instance of a `Provider` object given a string argument which contains the desired service name.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcp

Class ResourceUnavailableException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--jain.jcp.ResourceUnavailableException
```

public class **ResourceUnavailableException**

extends java.lang.Exception

This exception indicates that a resource inside the system is not available to complete an operation. The type embodied in this exception clarifies what is not available and is obtained via the getType() method in this class.

See Also:

Serialized Form

Field Summary

static int	**NO_DIALTONE** No dialtone detected.
static int	**OBSERVER_LIMIT_EXCEEDED** The number of observers existing already reached the limit.
static int	**ORIGINATOR_UNAVAILABLE** The originating device was not available for this action.
static int	**OUTSTANDING_METHOD_EXCEEDED** The internal resources to handle another method have been exceeded.
static int	**TRUNK_LIMIT_EXCEEDED** The number of trunks which are currently in use has been exceeded.
static int	**UNKNOWN** Indicates the specific reason is unspecified.

static int	**UNSPECIFIED_LIMIT_EXCEEDED** An internal resource, unspecified by the implementation, has been exceeded.
static int	**USER_RESPONSE** A user has not responded in the time allowed by an implementation.

Constructor Summary

ResourceUnavailableException(int type)
Constructor, takes a type but no string.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

UNKNOWN

public static final int **UNKNOWN**

Indicates the specific reason is unspecified.

ORIGINATOR_UNAVAILABLE

public static final int **ORIGINATOR_UNAVAILABLE**

The originating device was not available for this action.

OBSERVER_LIMIT_EXCEEDED

`public static final int` **`OBSERVER_LIMIT_EXCEEDED`**

The number of observers existing already reached the limit.

TRUNK_LIMIT_EXCEEDED

`public static final int` **`TRUNK_LIMIT_EXCEEDED`**

The number of trunks which are currently in use has been exceeded.

OUTSTANDING_METHOD_EXCEEDED

`public static final int` **`OUTSTANDING_METHOD_EXCEEDED`**

The internal resources to handle another method have been exceeded.

UNSPECIFIED_LIMIT_EXCEEDED

`public static final int` **`UNSPECIFIED_LIMIT_EXCEEDED`**

An internal resource, unspecified by the implementation, has been exceeded.

NO_DIALTONE

`public static final int` **`NO_DIALTONE`**

No dialtone detected.

USER_RESPONSE

`public static final int` **`USER_RESPONSE`**

A user has not responded in the time allowed by an implementation.

Constructor Detail

ResourceUnavailableException

`public` **`ResourceUnavailableException`**`(int type)`

Constructor, takes a type but no string.

---

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Class jain.jcp.ResourceUnavailableException

Packages that use ResourceUnavailableException

jain.jcc	
jain.jcp	

Uses of ResourceUnavailableException in jain.jcc

Methods in jain.jcc that throw ResourceUnavailableException

void	**JccProvider.addProviderListener**(JcpProviderListener providerlistener, EventFilter filter) Adds a listener to this provider.
void	**JccProvider.addCallListener**(JcpCallListener calllistener) Add a call listener to all call objects that will be created under the domain of this provider.
void	**JccProvider.addCallListener**(JcpCallListener calllistener, EventFilter filter) Add a call listener to all call objects that will be created under the domain of this provider.
void	**JccProvider.addCallLoadControlListener**(CallLoadControlListener loadcontrollistener, EventFilter filter) Adds a listener to listen to load control related events.
void	**JccConnection.routeConnection**(boolean attachmedia) Routes this JccConnection to the specified target address.
void	**JccConnection.release**() Drops a JccConnection from an active telephone call.
void	**JccCall.addCallListener**(JcpCallListener calllistener, EventFilter filter) Add a listener to this call.
void	**JccCall.release**() This method requests the release of the call object and associated connection objects.
JcpConnection	**JccCall.createConnection**(java.lang.String targetAddress, java.lang.String originatingAddress, java.lang.String originalCalledAddress, java.lang.String redirectingAddress) Creates a new JccConnection and attaches it to this JccCall.
JcpConnection	**JccCall.routeCall**(java.lang.String targetAddress, java.lang.String originatingAddress, java.lang.String originalDestinationAddress, java.lang.String redirectingAddress) This method requests routing of a call to the given call party.

Uses of ResourceUnavailableException in jain.jcp

Methods in jain.jcp that throw ResourceUnavailableException

JcpCall	**JcpProvider.createCall**() Creates a new instance of the call with no connections.

void	**JcpProvider.addProviderListener**(JcpProviderListener providerlistener) Adds a listener to this provider.
void	**JcpCall.addCallListener**(JcpCallListener calllistener) Add a listener to this call.

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

PREV NEXT FRAMES NO FRAMES

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

Uses of Interface jain.jcc.CallLoadControlEvent

Packages that use CallLoadControlEvent

jain.jcc	

Uses of CallLoadControlEvent in jain.jcc

Methods in jain.jcc with parameters of type CallLoadControlEvent	
`void`	**`CallLoadControlListener.providerCallOverloadEncountered`**`(CallLoadControlEvent loadcontrolevent)` This method indicates that the network has detected overload and may have automatically imposed load control on calls requested to a particular address range or calls made to a particular destination.
`void`	**`CallLoadControlListener.providerCallOverloadCeased`**`(CallLoadControlEvent loadcontrolevent)` This method indicates that the network has detected that the overload has ceased and has automatically removed load control on calls requested to a particular address range or calls made to a particular destination.

28 Aug 2000 If you have any comments or queries, please mail them to jcc@research.telcordia.com

jain.jcc

Interfaces

CallLoadControlEvent
CallLoadControlListener
EventFilter
JccAddress
JccCall
JccCallEvent
JccCallListener
JccConnection
JccConnectionEvent
JccConnectionListener
JccProvider

jain.jcp

Interfaces

Event
JcpAddress
JcpCall
JcpCallEvent
JcpCallListener
JcpConnection
JcpConnectionEvent
JcpConnectionListener
JcpPeer
JcpProvider
JcpProviderEvent
JcpProviderListener

Classes

JcpPeerFactory

Exceptions

InvalidArgumentException
InvalidPartyException
InvalidStateException
JcpPeerUnavailableException
MethodNotSupportedException
PlatformException
PrivilegeViolationException
ProviderUnavailableException
ResourceUnavailableException