Cisco Unified JTAPI Extensions

The Cisco Unified JTAPI extension consists of a set of classes and interfaces that expose the additional functionality not readily exposed in JTAPI 1.2 specification but are available in Cisco Unified Communications Manager. Developers can use the extensions to create new applications or modify existing extensions to create new methods.

This chapter describes the extensions (interfaces and classes) that are available for implementation in a Cisco Unified Communications Manager.

Class Hierarchy

The following class hierarchy is contained in the com.cisco.jtapi.extensions package.

```
hierarchy.java.lang.Object
   com.cisco.jtapi.extensions.CiscoAddressCallInfo
   com.cisco.jtapi.extensions.CiscoJtapiVersion
   com.cisco.jtapi.extensions.CiscoMediaCapability
      com.cisco.jtapi.extensions.CiscoG711MediaCapability
      com.cisco.jtapi.extensions.CiscoG723MediaCapability
      com.cisco.jtapi.extensions.CiscoG729MediaCapability
      com.cisco.jtapi.extensions.CiscoGSMMediaCapability
      com.cisco.jtapi.extensions.CiscoWideBandMediaCapability
   com.cisco.jtapi.extensions.CiscoRTPParams
   java.lang.Throwable (implements java.io.Serializable)
      java.lang.Exception
    com.cisco.jtapi.extensions.CiscoRegistrationException
    com.cisco.jtapi.extensions.CiscoUnregistrationException
```

CiscoAddressCallInfo

Class History

Cisco Unified Communications Manager Release	Description
7.1 (2)	Added the history table to track changes.

Declaration

public class CiscoAddressCallInfo extends java.lang.Object

java.lang.Object

com.cisco.jtapi.extensions.CiscoAddressCallInfo

Constructors

CiscoAddressCallInfo (int inumActiveCalls, int imaxActiveCalls, int inumCallsOnHold, int imaxCallsOnHold)

CiscoAddressCallInfo (int inumActiveCalls, int imaxActiveCalls, int inumCallsOnHold, int imaxCallsOnHold, CiscoCall[] icalls)

Fields

None

Methods

Table 1: Methods in CiscoAddressCallInfo

Interface	Method	Description
CiscoCall[]	getCalls()	Returns the array of Cisco calls on the CiscoAddress.
CiscoCall[]	getTerminal()	Returns the terminal on which the address got activated (i.e. marked unrestricted)
int	getMaxActiveCalls()	Returns the maximum number of active calls supported on the CiscoAddress, as an integer.
int	getMaxCallsOnHold()	Returns the maximum number of calls that can be put on hold on the CiscoAddress, as an integer.
int	getNumActiveCalls()	Returns the number of active calls on the CiscoAddress, as an integer.
int	getNumCallsOnHold()	Returns the number of held calls on the CiscoAddress, as an integer.

Inherited Methods

From Class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Related Documentation

None

CiscoG711MediaCapability

The CiscoG711MediaCapability object specifies the properties for a G.711 encoded RTP stream. Applications that support G.711 media termination use this object to specify their preferred packet size when registering a CiscoMediaTerminal. The default packet size is thirty milliseconds.

Class History

Cisco Unified Communications Manager Release	Description
7.1(x)	Added history table to track changes.

Declaration

public class CiscoG711MediaCapability extends CiscoMediaCapability

java.lang.Object

com.cisco.jtapi.extensions.CiscoMediaCapability

com.cisco.jtapi.extensions.CiscoG711MediaCapability

Constructors

Table 2: Constructors in CiscoG711MediaCapability

Interface	Constructor	Description
public	CiscoG711MediaCapability(intrtpPacketFrameSize)	Constructs a CiscoG711MediaCapability.
public	CiscoG711MediaCapability()	Constructs a CiscoG711MediaCapability.

Fields

Table 3: Fields in CiscoG711MediaCapability

Interface	Field	Description
public static final int	FRAMESIZE_TWENTY_MILLISECOND_PACKET	RTP Packet Framesize: Twenty millisecond RTP packet.
public static final int	FRAMESIZE_THIRTY_MILLISECOND_PACKET	RTP Packet Framesize: Thirty millisecond RTP packet.
public static final int	FRAMESIZE_SIXTY_MILLISECOND_PACKET	RTP Packet Framesize: Sixty millisecond RTP packet.

Inherited Fields

From Class com.cisco.jtapi.extensions.CiscoMediaCapability

G711_64K_30_MILLISECONDS, G723_6K_30_MILLISECONDS, G729_30_MILLISECONDS, GSM_80_MILLISECONDS, WIDEBAND_256K_10_MILLISECONDS

Methods

None

Inherited Methods

From Class com.cisco.jtapi.extensions.CiscoMediaCapability

getMaxFramesPerPacket, getPayloadType, isSupported, toString

From Class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Related Documentation

See Constant Field Values.

CiscoG723MediaCapability

The CiscoG723MediaCapability object specifies the properties for a G.723 encoded RTP stream. Applications that support G.723 media termination use this object to specify their preferred packet size and bit rate when registering a CiscoMediaTerminal. The default packet size is thirty milliseconds and the default bit rate is 6.4k.

Class History

Cisco Unified Communications Manager Release	Description
7.1x	Added history table to track changes.

Declaration

public class CiscoG723MediaCapability extends CiscoMediaCapability

java.lang.Object

com.cisco.jtapi.extensions.CiscoMediaCapability

com.cisco.jtapi.extensions.CiscoG723MediaCapability

Constructors

Table 4: Constructors in CiscoG723MediaCapability

Interface	Constructor	Description
public	CiscoG723MediaCapability (intrtpPacketFrameSize, intbitRate)	Constructs a CiscoG723MediaCapability.

Fields

Table 5: Fields in CiscoG723MediaCapability

Interface	Field	Description
public static final int	FRAMESIZE_TWENTY_MILLISECOND_PACKET	RTP Packet Framesize: Twenty millisecond RTP packet.
public static final int	FRAMESIZE_THIRTY_MILLISECOND_PACKET	RTP Packet Framesize: Thirty millisecond RTP packet.
public static final int	FRAMESIZE_SIXTY_MILLISECOND_PACKET	RTP Packet Framesize: Sixty millisecond RTP packet.

Inherited Fields

From Class com.cisco.jtapi.extensions.CiscoMediaCapability

G711_64K_30_MILLISECONDS, G723_6K_30_MILLISECONDS, G729_30_MILLISECONDS, GSM_80_MILLISECONDS, WIDEBAND_256K_10_MILLISECONDS

Methods

Table 6: Methods in CiscoG723MediaCapability

Interface	Method	Description
public int	getBitRate()	Returns the bit rate specified by this capability object. Returns: a bit rate from the RTPBitRate interface.
public java.lang.String	toString()	Overwrites the Object.toString() method. Overrides: toString in class CiscoMediaCapability.

Inherited Methods

From Class com.cisco.jtapi.extensions.CiscoMediaCapability

getMaxFramesPerPacket, getPayloadType, isSupported

From Class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Related Documentation

See Constant Field Values.

CiscoG729MediaCapability

The CiscoG729MediaCapability object specifies the properties for a G.729 encoded RTP stream. Applications that support G.729 media termination use this object to specify their preferred packet size when registering a CiscoMediaTerminal. The default packet size is thirty milliseconds.

Class History

Cisco Unified Communications Manager Release	Description
7.1x	Added history table to track changes.

Declaration

public class CiscoG729MediaCapability extends CiscoMediaCapability

java.lang.Object

com.cisco.jtapi.extensions.CiscoMediaCapability

com.cisco.jtapi.extensions.CiscoG729MediaCapability

Constructors

Table 7: Constructors in G729MediaCapability

Constructor	Description
CiscoG729MediaCapability(int payload, int rtpPacketFrameSize)	Constructs a CiscoG729MediaCapability.

Fields

Table 8: Fields in CiscoG729MediaCapability

Interface	Fields	Description
staticint	FRAMESIZE_SIXTY_MILLISECOND_PACKET	RTP Packet Framesize: Sixty millisecond RTP packet.
staticint	FRAMESIZE_THIRTY_MILLISECOND_PACKET	RTP Packet Framesize: Thirty millisecond RTP packet.
staticint	FRAMESIZE_TWENTY_MILLISECOND_PACKET	RTP Packet Framesize: Twenty millisecond RTP packet.

Inherited Fields

From Class com.cisco.jtapi.extensions.CiscoMediaCapability

G711_64K_30_MILLISECONDS, G723_6K_30_MILLISECONDS, G729_30_MILLISECONDS, GSM_80_MILLISECONDS, WIDEBAND_256K_10_MILLISECONDS

Methods

None

Inherited Methods

From Class com.cisco.jtapi.extensions.CiscoMediaCapability

getMaxFramesPerPacket, getPayloadType, isSupported, toString

From Class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Related Documentation

See Constant Field Values.

CiscoGSMMediaCapability

The CiscoGSMMediaCapability object specifies the properties for a GSM encoded RTP stream. Applications that support GSM media termination use this object to specify their preferred packet size when registering a CiscoMediaTerminal. The default packet size is thirty milliseconds.

Class History

Cisco Unified Communications Manager Release	Description
7.1x	Added history table to track changes.

Declaration

public class CiscoGSMMediaCapability extends CiscoMediaCapability

java.lang.Object

com.cisco.jtapi.extensions.CiscoMediaCapability

com.cisco.jtapi.extensions.CiscoGSMMediaCapability

Constructors

Table 9: Constructors in CiscoGSMMediaCapability

Interface	Constructor	Description
public	CiscoGSMMediaCapability()	Constructs a CiscoGSMMediaCapability
public	CiscoGSMMediaCapability(int rtpPacketFrameSize)	Constructs a CiscoGSMMediaCapability.

Fields

Table 10: Fields in CiscoGSMMediaCapability

Interface	Field	Description
staticint	FRAMESIZE_EIGHTY_MILLISECOND_PACKET	RTP Packet Framesize: Eighty millisecond RTP packet

Inherited Fields

From Class com.cisco.jtapi.extensions.CiscoMediaCapability

G711_64K_30_MILLISECONDS, G723_6K_30_MILLISECONDS, G729_30_MILLISECONDS, GSM_80_MILLISECONDS, WIDEBAND_256K_10_MILLISECONDS

Methods

None

Inherited Methods

From Class com.cisco.jtapi.extensions.CiscoMediaCapability

getMaxFramesPerPacket, getPayloadType, isSupported, toString

From Class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Related Documentation

None

CiscoJtapiVersion

This class gives the version information of the installed Cisco JTAPI. Programs can get the version number using the accessor methods. Cisco Jtapi Version is in a.b(x.y) format where “a” indicates the major version, “b” indicates the minor version, “x” indicates the revision number, and “y” indicates the build number .

Class History

Cisco Unified Communications Manager Release	Description
7.1x	Added history table to track changes.

Declaration

public class CiscoJtapiVersion extends java.lang.Object

java.lang.Object

com.cisco.jtapi.extensions.CiscoJtapiVersion

Constructors

publicCiscoJtapiVersion()None

Fields

None

Methods

Table 11: Methods in CiscoJtapiVersion

Interface	Method	Description
java.lang.String	getBuildDescription()	Returns “release” if it is a release version or debug if it is not a release version.
int	getBuildNumber()	Returns the build number of the version.
int	getExtendedBuildNumber()	Returns the extended build number of the version.
int	getMajorVersion()	Returns the major version number.
int	getMinorVersion()	Returns the minor version number.
int	getRevisionNumber()	Returns the revision number of the version.
public java.lang.String	getVersion()	Returns the version information in a.b(x.y)-z format without a name.
public java.lang.String	toString()	Returns the version information in a.b(x.y)-z format. Overrides toString in class java.lang.Object.

Inherited Methods

From Class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Related Documentation

None

CiscoMediaCapability

The CiscoMediaCapability object specifies the properties of a particular media format that an application can support for CiscoMediaTerminals that it registers. Because CiscoMediaCapability is an abstract class, applications may only construct its subclasses directly.

Class History

Cisco Unified Communications Manager Release	Description
7.1x	Added history table to track changes.

Declaration

public class CiscoMediaCapability extends java.lang.Object

java.lang.Object

com.cisco.jtapi.extensions.CiscoMediaCapability

Subclasses

CiscoG711MediaCapability, CiscoG723MediaCapability, CiscoG729MediaCapability, CiscoGSMMediaCapability, CiscoWideBandMediaCapability

Constructors

Table 12: Constructors in CiscoMediaCapability

Interface	Constructor	Description
public	CiscoMediaCapability(intpayloadType, intmaxFramesPerPacket)	Constructs a CiscoMediaCapability object for the specified payload type and packet size (in milliseconds).

Fields

Table 13: Fields in CiscoMediaCapability

Interface	Field	Description
static	G711_64K_30_MILLISECONDS	G.711 capability with default parameters.
static	G723_6K_30_MILLISECONDS	G.723 capability with default parameters.
static	G729_30_MILLISECONDS	G.729 capability with default parameters.
static	GSM_80_MILLISECONDS	GSM capability with default parameters.
static	WIDEBAND_256K_10_MILLISECONDS	Wideband capability with default parameters.

Methods

Table 14: Methods in CiscoMediaCapability

Interface	Method	Description
int	getMaxFramesPerPacket(	Returns the packet size (in milliseconds) that this object specifies.The maxFramesPerPacket parameter is a carryover from the H.245 protocol definition. Cisco Unified Communications Manager does not use this field as the number of frames per RTP packet, but rather as the number of milliseconds of audio per RTP packet that the device can receive. Third-party IP phones may use different (higher) rates even though these rates may not be exceeded to and or from Cisco Unified IP phones.
int	getPayloadType()	Returns a payload type from the RTPPayload interface that this object specifies.
boolean	isSupported()	Returns whether the payload of this object is supported or not. True if the payloadType is supported, or otherwise false
java.lang.String	toString()	Overrides toString in class java.lang.Object.

Inherited Methods

From Class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Related Documentation

See CiscoG711MediaCapability, CiscoG723MediaCapability, CiscoG729MediaCapability, CiscoGSMMediaCapability, CiscoWideBandMediaCapability, CiscoRTPBitRate, and CiscoRTPPayload.

CiscoMultiMediaCapabilityInfo

CiscoMultiMediaCapabilityInfo interface contains the multimedia capabilities of a terminal. Applications can get the video capability, number of screens, and telepresence interoperability of the terminal using this API.

Declaration

public interface CiscoMultiMediaCapabilityInfo

com.cisco.jtapi.extensions.CiscoMultiMediaCapabilityInfo

Fields

Table 15: Fields in CiscoMultiMediaCapabilityInfo

Interface	Field	Description
static final int	NONE	Indicates that the CiscoMultiMediaCapabilityInfo.getVideoCapability () for this terminal is NONE.
static final int	VIDEO_ENABLED	CiscoMultiMediaCapabilityInfo.getVideoCapability () for this terminal is VIDEO_ENABLED.
static final int	TELEPRESENCEINTEROP_NONE	Indicates that the CiscoMultiMediaCapabilityInfo.getTelepresenceInfo() for this terminal is TELEPRESENCEINTEROP_NONE.
static final int	TELEPRESENCEINTEROP_ENABLED	CiscoMultiMediaCapabilityInfo. getTelepresenceInfo () for this terminal is TELEPRESENCEINTEROP_ENABLED

Methods

Table 16: Methods in MultiMediaCapabilityInfo

Interface	Method	Description
int	getVideoCapability()	Returns the video capability of the Terminal. The video capability can be NONE or VIDEO_ENABLED
int	getTelepresenceInfo()	Returns the telepresence capability of the Terminal. The telepresence capability can be TELEPRESENCEINTEROP_NONE or TELEPRESENCEINTEROP_ENABLED
int	getScreenCount()	Returns the number of screens present on the Terminal.

CiscoRegistrationException

The CiscoMediaTerminal.register method throws this exception when the registration process fails for any reason. For example, registration would fail if the Provider were OUT_OF_SERVICE or if the device were already registered.

Class History

Cisco Unified Communications Manager Release	Description
7.1x	Added history table to track changes.

Declaration

public class CiscoRegistrationException extends java.lang.Exception

java.lang.Object

java.lang.Throwable

java.lang.Exception

com.cisco.jtapi.extensions.CiscoRegistrationException

Implemented Interfaces

java.io.Serializable

Constructors

Table 17: Constructors in CiscoRegistrationException

Interface	Constructor	Description
public	CiscoRegistrationException (java.lang.Stringdescription)	Takes the description of the exception as a parameter.

Methods

None

Inherited Methods

From Class java.lang.Throwable

fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

From Class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Related Documentation

See CiscoMediaTerminal.register(java.net.InetAddress, int, com.cisco.jtapi.extensions.CiscoMediaCapability[]).

CiscoRTPParams

You can use the CiscoRTPParams class to specify a dynamic RTP address and port number for a media terminal on a per-call basis. Applications can pass this object in setRTPParams() of CiscoMediaTerminal. These parameters are only valid for a particular call.

Class History

Cisco Unified Communications Manager Release	Description
7.1x	Added history table to track changes.

Declaration

public class CiscoRTPParams extends java.lang.Object

java.lang.Object

Constructors

CiscoRTPParams (java.net.InetAddress, rtpAddress, int rtpPort)

Fields

None

Methods

Table 18: Methods in CiscoRTPParams

Interface	Method	Description
java.net.InetAddress	getRTPAddress()	Returns the Internet address for the inbound RTP stream of the associated call.
java.lang.String	getRTPAddressHostName()	Returns the IP host name for the inbound RTP stream of the associated call.
byte[]	getRTPByteAddress()	Returns the Internet address in byte format for the inbound RTP stream.
int	getRTPPort()	Returns the UDP port for the inbound RTP stream.
java.lang.String	toString()	Returns a String in the format “IP address/port number.” Overrides toString in class java.lang.Object.

Inherited Methods

From Class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Related Documentation

See CiscoTerminal and CiscoMediaTerminal.

CiscoUnregistrationException

The CiscoMediaTerminal.unregister method throws this exception when the unregistration process fails. For example, registration fails if the Provider is OUT_OF_SERVICE or the Terminal is already unregistered.

Class History

Cisco Unified Communications Manager Release	Description
7.1x	Added history table to track changes.

Declaration

public class CiscoUnregistrationException extends java.lang.Exception

java.lang.Object

java.lang.Throwable

java.lang.Exception

com.cisco.jtapi.extensions.CiscoUnregistrationException

Implemented Interfaces

java.io.Serializable

Constructors

Table 19: Constructors in CiscoUnregistrationException

Interface	Constructor	Description
public	CiscoUnregistrationException ()(java.lang.Stringdescription)	None

Fields

None

Methods

None

Inherited Methods

From Class java.lang.Throwable

fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

From Class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Related Documentation

See CiscoMediaTerminal.unregister(), Serialized Form.

CiscoWideBandMediaCapability

The CiscoWideBandMediaCapability object specifies the properties for a wide band encoded RTP stream. Applications that support wide band media termination use this object to specify their preferred packet size when registering a CiscoMediaTerminal. The default packet size is ten milliseconds.

Class History

Cisco Unified Communications Manager Release	Description
7.1x	Added history table to track changes.

Declaration

public class CiscoWideBandMediaCapability extends CiscoMediaCapability

java.lang.Object

com.cisco.jtapi.extensions.CiscoMediaCapability

com.cisco.jtapi.extensions.CiscoWideBandMediaCapability

Constructors

Table 20: Constructors in CiscoWideBandMediaCapability

Interface	Constructor	Description
public	CiscoWideBandMediaCapability(intpacketsize)	Constructs a CiscoWideBandMediaCapability object with the specified packet size. The default is ten–millisecond packet size. **Parameters** • packetsize—The RTP packet Framesize.

Fields

Table 21: Fields in CiscoWideBandMedicaCapability

Interface	Field	Description
staticint	FRAMESIZE_TEN_MILLISECOND_PACKET	RTP Packet Framesize: Ten millisecond RTP packet

Inherited Fields

From Class com.cisco.jtapi.extensions.CiscoMediaCapability

G711_64K_30_MILLISECONDS, G723_6K_30_MILLISECONDS, G729_30_MILLISECONDS, GSM_80_MILLISECONDS, WIDEBAND_256K_10_MILLISECONDS

Methods

None

Inherited Methods

From Class com.cisco.jtapi.extensions.CiscoMediaCapability

getMaxFramesPerPacket, getPayloadType, isSupported, toString

From Class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Related Documentation

See Constant Field Values.

Interface Hierarchy

The following interface hierarchy is contained in the com.cisco.jtapi.extensions package hierarchy.

```
javax.telephony.Address
   com.cisco.jtapi.extensions.CiscoAddress (also extends
                                com.cisco.jtapi.extensions.CiscoObjectContainer)
      com.cisco.jtapi.extensions.CiscoIntercomAddress
   javax.telephony.callcenter.RouteAddress
      com.cisco.jtapi.extensions.CiscoRouteAddress


javax.telephony.AddressObserver
   com.cisco.jtapi.extensions.CiscoAddressObserver


javax.telephony.Call
   javax.telephony.callcontrol.CallControlCall
      com.cisco.jtapi.extensions.CiscoCall (also extends
                                com.cisco.jtapi.extensions.CiscoObjectContainer)
   com.cisco.jtapi.extensions.CiscoConsultCall


com.cisco.jtapi.extensions.CiscoCallCtlTermConnHeldReversionEv


com.cisco.jtapi.extensions.CiscoConferenceChain


com.cisco.jtapi.extensions.CiscoFeatureReason


com.cisco.jtapi.extensions.CiscoJtapiException


com.cisco.jtapi.extensions.CiscoJtapiProperties


com.cisco.jtapi.extensions.CiscoLocales


com.cisco.jtapi.extensions.CiscoMediaSecurityIndicator


com.cisco.jtapi.extensions.CiscoMediaConnectionMode


com.cisco.jtapi.extensions.CiscoMediaEncryptionAlgorithmType


com.cisco.jtapi.extensions.CiscoMediaEncryptionKeyInfo


com.cisco.jtapi.extensions.CiscoMediaSecurityIndicator


com.cisco.jtapi.extensions.CiscoMonitorInitiatorInfo


com.cisco.jtapi.extensions.CiscoMonitorTargetInfo


com.cisco.jtapi.extensions.CiscoObjectContainer
   com.cisco.jtapi.extensions.CiscoAddress (also extends javax.telephony.Address)
     com.cisco.jtapi.extensions.CiscoIntercomAddress
   com.cisco.jtapi.extensions.CiscoCall (also extends
                                        javax.telephony.callcontrol.CallControlCall)
```

```
   com.cisco.jtapi.extensions.CiscoConsultCall
   com.cisco.jtapi.extensions.CiscoCallID
   com.cisco.jtapi.extensions.CiscoConnection (also extends
                              javax.telephony.callcontrol.CallControlConnection)
   com.cisco.jtapi.extensions.CiscoConnectionID
   com.cisco.jtapi.extensions.CiscoConsultCall
   com.cisco.jtapi.extensions.CiscoIntercomAddress
   com.cisco.jtapi.extensions.CiscoJtapiPeer (also extends javax.telephony.JtapiPeer,
                                         com.cisco.services.tracing.TraceModule)
   com.cisco.jtapi.extensions.CiscoMediaTerminal
   com.cisco.jtapi.extensions.CiscoProvider
   com.cisco.jtapi.extensions.CiscoRouteTerminal
   com.cisco.jtapi.extensions.CiscoTerminal (also extends javax.telephony.Terminal)
      com.cisco.jtapi.extensions.CiscoMediaTerminal
      com.cisco.jtapi.extensions.CiscoRouteTerminal
   com.cisco.jtapi.extensions.CiscoTerminalConnection (also extends
                       javax.telephony.callcontrol.CallControlTerminalConnection)

com.cisco.jtapi.extensions.CiscoPartyInfo

com.cisco.jtapi.extensions.CiscoProvFeatureID

com.cisco.jtapi.extensions.CiscoProviderCapabilityChangedEv

com.cisco.jtapi.extensions.CiscoRecorderInfo

com.cisco.jtapi.extensions.CiscoRTPBitRate

com.cisco.jtapi.extensions.CiscoRTPHandle

com.cisco.jtapi.extensions.CiscoRTPInputProperties

com.cisco.jtapi.extensions.CiscoRTPOutputProperties

com.cisco.jtapi.extensions.CiscoRTPPayload

com.cisco.jtapi.extensions.CiscoSynchronousObserver

com.cisco.jtapi.extensions.CiscoTermConnPrivacyChangedEv

com.cisco.jtapi.extensions.CiscoTermEvFilter

com.cisco.jtapi.extensions.CiscoTerminalProtocol

com.cisco.jtapi.extensions.CiscoTone

com.cisco.jtapi.extensions.CiscoUrlInfo

javax.telephony.Connection
   javax.telephony.callcontrol.CallControlConnection
      com.cisco.jtapi.extensions.CiscoConnection (also extends
                                  com.cisco.jtapi.extensions.CiscoObjectContainer)

javax.telephony.events.Ev
```

```
   javax.telephony.events.AddrEv
      com.cisco.jtapi.extensions.CiscoAddrEv (also extends
                                                  com.cisco.jtapi.extensions.CiscoEv)
       com.cisco.jtapi.extensions.CiscoAddrAutoAcceptStatusChangedEv
       com.cisco.jtapi.extensions.CiscoAddrInServiceEv
       com.cisco.jtapi.extensions.CiscoAddrIntercomInfoChangedEv
       com.cisco.jtapi.extensions.CiscoAddrIntercomInfoRestorationFailedEv
       com.cisco.jtapi.extensions.CiscoAddrOutOfServiceEv (also extends
                                com.cisco.jtapi.extensions.CiscoOutOfServiceEv)
       com.cisco.jtapi.extensions.CiscoAddressRecordingConfigChangedEv
   javax.telephony.callcontrol.events.CallCtlEv
      javax.telephony.callcontrol.events.CallCtlCallEv (also extends
                                                          javax.telephony.events.CallEv)
       javax.telephony.callcontrol.events.CallCtlConnEv (also extends
                                              javax.telephony.events.ConnEv)
  javax.telephony.callcontrol.events.CallCtlConnOfferedEv
com.cisco.jtapi.extensions.CiscoCallCtlConnOfferedEv
   javax.telephony.events.CallEv
      javax.telephony.events.CallActiveEv
       com.cisco.jtapi.extensions.CiscoConsultCallActiveEv (also extends
                                  com.cisco.jtapi.extensions.CiscoCallEv)
      javax.telephony.callcontrol.events.CallCtlCallEv (also extends
                                javax.telephony.callcontrol.events.CallCtlEv)
       javax.telephony.callcontrol.events.CallCtlConnEv (also extends
                                              javax.telephony.events.ConnEv)
   javax.telephony.callcontrol.events.CallCtlConnOfferedEv
   com.cisco.jtapi.extensions.CiscoCallCtlConnOfferedEv
       com.cisco.jtapi.extensions.CiscoCallEv (also extends
                                                  com.cisco.jtapi.extensions.CiscoEv)
       com.cisco.jtapi.extensions.CiscoCallChangedEv
       com.cisco.jtapi.extensions.CiscoCallSecurityStatusChangedEv
       com.cisco.jtapi.extensions.CiscoConferenceChainAddedEv
       com.cisco.jtapi.extensions.CiscoConferenceChainRemovedEv
       com.cisco.jtapi.extensions.CiscoConferenceEndEv
       com.cisco.jtapi.extensions.CiscoConferenceStartEv
       com.cisco.jtapi.extensions.CiscoConsultCallActiveEv (also extends
                                      javax.telephony.events.CallActiveEv)
       com.cisco.jtapi.extensions.CiscoToneChangedEv
       com.cisco.jtapi.extensions.CiscoTransferEndEv
       com.cisco.jtapi.extensions.CiscoTransferStartEv
   javax.telephony.events.ConnEv
      javax.telephony.callcontrol.events.CallCtlConnEv (also extends
                                javax.telephony.callcontrol.events.CallCtlCallEv)
   javax.telephony.callcontrol.events.CallCtlConnOfferedEv
   com.cisco.jtapi.extensions.CiscoCallCtlConnOfferedEv
   javax.telephony.events.TermConnEv
      com.cisco.jtapi.extensions.CiscoTermConnMonitoringEndEv
      com.cisco.jtapi.extensions.CiscoTermConnMonitoringStartEv
      com.cisco.jtapi.extensions.CiscoTermConnMonitorInitiatorInfoEv
      com.cisco.jtapi.extensions.CiscoTermConnMonitorTargetInfoEv
      com.cisco.jtapi.extensions.CiscoTermConnRecordingEndEv
      com.cisco.jtapi.extensions.CiscoTermConnRecordingStartEv
      com.cisco.jtapi.extensions.CiscoTermConnRecordingTargetInfoEv
      com.cisco.jtapi.extensions.CiscoTermConnSelectChangedEv


com.cisco.jtapi.extensions.CiscoEv
   com.cisco.jtapi.extensions.CiscoAddrActivatedEv
   com.cisco.jtapi.extensions.CiscoAddrActivatedOnTerminalEv
   com.cisco.jtapi.extensions.CiscoAddrAddedToTerminalEv
   com.cisco.jtapi.extensions.CiscoAddrAutoAcceptStatusChangedEv
   com.cisco.jtapi.extensions.CiscoAddrCreatedEv
   com.cisco.jtapi.extensions.CiscoAddrEv (also extends
                                                          javax.telephony.events.AddrEv)
      com.cisco.jtapi.extensions.CiscoAddrAutoAcceptStatusChangedEv
```

```
      com.cisco.jtapi.extensions.CiscoAddrInServiceEv
      com.cisco.jtapi.extensions.CiscoAddrIntercomInfoChangedEv
      com.cisco.jtapi.extensions.CiscoAddrIntercomInfoRestorationFailedEv
      com.cisco.jtapi.extensions.CiscoAddrOutOfServiceEv (also extends
                                    com.cisco.jtapi.extensions.CiscoAddrEv,
                                com.cisco.jtapi.extensions.CiscoOutOfServiceEv)
      com.cisco.jtapi.extensions.CiscoAddressRecordingConfigChangedEv
      com.cisco.jtapi.extensions.CiscoAddrInServiceEv
      com.cisco.jtapi.extensions.CiscoAddrIntercomInfoChangedEv
      com.cisco.jtapi.extensions.CiscoAddrIntercomInfoRestorationFailedEv
      com.cisco.jtapi.extensions.CiscoAddrOutOfServiceEv (also extends
                                     com.cisco.jtapi.extensions.CiscoAddrEv)
      com.cisco.jtapi.extensions.CiscoAddressRecordingConfigChangedEv
   com.cisco.jtapi.extensions.CiscoAddrRemovedEv
   com.cisco.jtapi.extensions.CiscoAddrRemovedFromTerminalEv
   com.cisco.jtapi.extensions.CiscoAddrRestrictedEv
   com.cisco.jtapi.extensions.CiscoAddrRestrictedOnTerminalEv
   com.cisco.jtapi.extensions.CiscoCallChangedEv
   com.cisco.jtapi.extensions.CiscoCallEv (also extends
                                                        javax.telephony.events.CallEv)
      com.cisco.jtapi.extensions.CiscoCallChangedEv
      com.cisco.jtapi.extensions.CiscoCallSecurityStatusChangedEv
      com.cisco.jtapi.extensions.CiscoConferenceChainAddedEv
      com.cisco.jtapi.extensions.CiscoConferenceChainRemovedEv
      com.cisco.jtapi.extensions.CiscoConferenceEndEv
      com.cisco.jtapi.extensions.CiscoConferenceStartEv
      com.cisco.jtapi.extensions.CiscoConsultCallActiveEv (also extends
                                          javax.telephony.events.CiscoCallEv)
      com.cisco.jtapi.extensions.CiscoToneChangedEv
      com.cisco.jtapi.extensions.CiscoTransferEndEv
      com.cisco.jtapi.extensions.CiscoTransferStartEv


com.cisco.jtapi.extensions.CiscoCallSecurityStatusChangedEv


com.cisco.jtapi.extensions.CiscoConferenceChainAddedEv


com.cisco.jtapi.extensions.CiscoConferenceChainRemovedEv


com.cisco.jtapi.extensions.CiscoConferenceEndEv


com.cisco.jtapi.extensions.CiscoConferenceStartEv


com.cisco.jtapi.extensions.CiscoConsultCallActiveEv (also extends
                                        javax.telephony.events.CallActiveEv,
                                           com.cisco.jtapi.extensions.CiscoCallEv)


com.cisco.jtapi.extensions.CiscoMediaOpenLogicalChannelEv


com.cisco.jtapi.extensions.CiscoOutOfServiceEv
   com.cisco.jtapi.extensions.CiscoAddrOutOfServiceEv (also extends
                                     com.cisco.jtapi.extensions.CiscoAddrEv)
   com.cisco.jtapi.extensions.CiscoTermOutOfServiceEv (also extends
                                     com.cisco.jtapi.extensions.CiscoTermEv)


com.cisco.jtapi.extensions.CiscoProvCallParkEv


com.cisco.jtapi.extensions.CiscoProvFeatureEv (also extends
                                                        javax.telephony.events.ProvEv)
```

```
com.cisco.jtapi.extensions.CiscoAddrActivatedEv
com.cisco.jtapi.extensions.CiscoAddrActivatedOnTerminalEv
com.cisco.jtapi.extensions.CiscoAddrAddedToTerminalEv
com.cisco.jtapi.extensions.CiscoAddrCreatedEv
com.cisco.jtapi.extensions.CiscoAddrRemovedEv
com.cisco.jtapi.extensions.CiscoAddrRemovedFromTerminalEv
com.cisco.jtapi.extensions.CiscoAddrRestrictedEv
com.cisco.jtapi.extensions.CiscoAddrRestrictedOnTerminalEv
com.cisco.jtapi.extensions.CiscoProvCallParkEv
com.cisco.jtapi.extensions.CiscoProvFeatureEv
   com.cisco.jtapi.extensions.CiscoProvCallParkEv
com.cisco.jtapi.extensions.CiscoRestrictedEv
   com.cisco.jtapi.extensions.CiscoAddrRestrictedEv
   com.cisco.jtapi.extensions.CiscoAddrRestrictedOnTerminalEv
   com.cisco.jtapi.extensions.CiscoTermActivatedEv
   com.cisco.jtapi.extensions.CiscoTermCreatedEv
   com.cisco.jtapi.extensions.CiscoTermRemovedEv
   com.cisco.jtapi.extensions.CiscoTermRestrictedEv
   com.cisco.jtapi.extensions.CiscoProvFeatureEv
   com.cisco.jtapi.extensions.CiscoProvCallParkEv
com.cisco.jtapi.extensions.CiscoRestrictedEv
   com.cisco.jtapi.extensions.CiscoAddrRestrictedEv
   com.cisco.jtapi.extensions.CiscoAddrRestrictedOnTerminalEv
com.cisco.jtapi.extensions.CiscoRTPInputKeyEv
com.cisco.jtapi.extensions.CiscoRTPInputStartedEv
com.cisco.jtapi.extensions.CiscoRTPInputStoppedEv
com.cisco.jtapi.extensions.CiscoRTPOutputKeyEv
com.cisco.jtapi.extensions.CiscoRTPOutputStartedEv
com.cisco.jtapi.extensions.CiscoRTPOutputStoppedEv
com.cisco.jtapi.extensions.CiscoTermActivatedEv
com.cisco.jtapi.extensions.CiscoTermButtonPressedEv
com.cisco.jtapi.extensions.CiscoTermCreatedEv
com.cisco.jtapi.extensions.CiscoTermDataEv
com.cisco.jtapi.extensions.CiscoTermDeviceStateActiveEv
com.cisco.jtapi.extensions.CiscoTermDeviceStateAlertingEv
com.cisco.jtapi.extensions.CiscoTermDeviceStateHeldEv
com.cisco.jtapi.extensions.CiscoTermDeviceStateWhisperEv
com.cisco.jtapi.extensions.CiscoTermDeviceStateWhisperEv
com.cisco.jtapi.extensions.CiscoTermDNDStatusChangedEv
com.cisco.jtapi.extensions.CiscoTermEvFilter (also extends
                                                 javax.telephony.events.TermEv)
   com.cisco.jtapi.extensions.CiscoMediaOpenLogicalChannelEv
   com.cisco.jtapi.extensions.CiscoRTPInputKeyEv
   com.cisco.jtapi.extensions.CiscoRTPInputStartedEv
   com.cisco.jtapi.extensions.CiscoRTPInputStoppedEv
   com.cisco.jtapi.extensions.CiscoRTPOutputKeyEv
   com.cisco.jtapi.extensions.CiscoRTPOutputStartedEv
   com.cisco.jtapi.extensions.CiscoRTPOutputStoppedEv
   com.cisco.jtapi.extensions.CiscoTermButtonPressedEv
   com.cisco.jtapi.extensions.CiscoTermDataEv
   com.cisco.jtapi.extensions.CiscoTermDeviceStateActiveEv
   com.cisco.jtapi.extensions.CiscoTermDeviceStateAlertingEv
   com.cisco.jtapi.extensions.CiscoTermDeviceStateHeldEv
   com.cisco.jtapi.extensions.CiscoTermDeviceStateIdleEv
   com.cisco.jtapi.extensions.CiscoTermDeviceStateWhisperEv
   com.cisco.jtapi.extensions.CiscoTermDNDStatusChangedEv
   com.cisco.jtapi.extensions.CiscoTermInServiceEv
   com.cisco.jtapi.extensions.CiscoTermOutOfServiceEv(also extends
                           com.cisco.jtapi.extensions.CiscoOutOfServiceEv)
   com.cisco.jtapi.extensions.CiscoTermRegistrationFailedEv
   com.cisco.jtapi.extensions.CiscoTermSnapshotCompletedEv
   com.cisco.jtapi.extensions.CiscoTermSnapshotEv
com.cisco.jtapi.extensions.CiscoTermInServiceEv
com.cisco.jtapi.extensions.CiscoTermOutOfServiceEv (also extends
```

```
                                com.cisco.jtapi.extensions.CiscoOutOfServiceEv,
                                                    com.cisco.jtapi.extensions.CiscoTermEv)
   com.cisco.jtapi.extensions.CiscoTermRegistrationFailedEv
   com.cisco.jtapi.extensions.CiscoTermRemovedEv
   com.cisco.jtapi.extensions.CiscoTermRestrictedEv
   com.cisco.jtapi.extensions.CiscoTermSnapshotCompletedEv
   com.cisco.jtapi.extensions.CiscoTermSnapshotEv
   com.cisco.jtapi.extensions.CiscoToneChangedEv
   com.cisco.jtapi.extensions.CiscoTransferEndEv
   com.cisco.jtapi.extensions.CiscoTransferStartEv


javax.telephony.events.ProvEv
   com.cisco.jtapi.extensions.CiscoProvEv (also extends
                                                    com.cisco.jtapi.extensions.CiscoEv)
      com.cisco.jtapi.extensions.CiscoAddrActivatedEv
      com.cisco.jtapi.extensions.CiscoAddrActivatedOnTerminalEv
      com.cisco.jtapi.extensions.CiscoAddrAutoAcceptStatusChangedEv
      com.cisco.jtapi.extensions.CiscoAddrCreatedEv
      com.cisco.jtapi.extensions.CiscoAddrRemovedEv
      com.cisco.jtapi.extensions.CiscoAddrRemovedFromTerminalEv
      com.cisco.jtapi.extensions.CiscoAddrRestrictedEv
      com.cisco.jtapi.extensions.CiscoAddrRestrictedOnTerminalEv
      com.cisco.jtapi.extensions.CiscoProvCallParkEv
      com.cisco.jtapi.extensions.CiscoProvFeatureEv
    com.cisco.jtapi.extensions.CiscoProvCallParkEv
      com.cisco.jtapi.extensions.CiscoRestrictedEv
    com.cisco.jtapi.extensions.CiscoAddrRestrictedEv
    com.cisco.jtapi.extensions.CiscoAddrRestrictedOnTerminalEv
      com.cisco.jtapi.extensions.CiscoTermActivatedEv
      com.cisco.jtapi.extensions.CiscoTermCreatedEv
      com.cisco.jtapi.extensions.CiscoTermRemovedEv
      com.cisco.jtapi.extensions.CiscoTermRestrictedEv


javax.telephony.events.TermEv
   com.cisco.jtapi.extensions.CiscoTermEv (also extends
                                                    com.cisco.jtapi.extensions.CiscoEv)
   com.cisco.jtapi.extensions.CiscoMediaOpenLogicalChannelEv
   com.cisco.jtapi.extensions.CiscoRTPInputKeyEv
   com.cisco.jtapi.extensions.CiscoRTPInputStartedEv
   com.cisco.jtapi.extensions.CiscoRTPInputStoppedEv
   com.cisco.jtapi.extensions.CiscoRTPOutputKeyEv
   com.cisco.jtapi.extensions.CiscoRTPOutputStartedEv
   com.cisco.jtapi.extensions.CiscoRTPOutputStoppedEv
   com.cisco.jtapi.extensions.CiscoTermButtonPressedEv
   com.cisco.jtapi.extensions.CiscoTermDataEv
   com.cisco.jtapi.extensions.CiscoTermDeviceStateActiveEv
   com.cisco.jtapi.extensions.CiscoTermDeviceStateAlertingEv
   com.cisco.jtapi.extensions.CiscoTermDeviceStateHeldEv
   com.cisco.jtapi.extensions.CiscoTermDeviceStateIdleEv
   com.cisco.jtapi.extensions.CiscoTermDeviceStateWhisperEv
   com.cisco.jtapi.extensions.CiscoTermDNDStatusChangedEv
   com.cisco.jtapi.extensions.CiscoTermInServiceEv
   com.cisco.jtapi.extensions.CiscoTermOutOfServiceEv (also extends
                                       com.cisco.jtapi.extensions.CiscoOutOfServiceEv)
   com.cisco.jtapi.extensions.CiscoTermRegistrationFailedEv
   com.cisco.jtapi.extensions.CiscoTermSnapshotCompletedEv
   com.cisco.jtapi.extensions.CiscoTermSnapshotEv


javax.telephony.JtapiPeer
   com.cisco.jtapi.extensions.CiscoJtapiPeer (also extends
                               com.cisco.jtapi.extensions.CiscoObjectContainer,
                                                    com.cisco.services.tracing.TraceModule)
```

```
javax.telephony.Provider
   com.cisco.jtapi.extensions.CiscoProvider (also extends
                                com.cisco.jtapi.extensions.CiscoObjectContainer)


javax.telephony.capabilities.ProviderCapabilities
   com.cisco.jtapi.extensions.CiscoProviderCapabilities


javax.telephony.ProviderObserver
   com.cisco.jtapi.extensions.CiscoProviderObserver


javax.telephony.callcenter.RouteSession
   com.cisco.jtapi.extensions.CiscoRouteSession


javax.telephony.callcenter.events.RouteSessionEvent
   javax.telephony.callcenter.events.RouteEvent
      com.cisco.jtapi.extensions.CiscoRouteEvent
   javax.telephony.callcenter.events.RouteUsedEvent
      com.cisco.jtapi.extensions.CiscoRouteUsedEvent


javax.telephony.Terminal
   com.cisco.jtapi.extensions.CiscoTerminal (also extends
                           com.cisco.jtapi.extensions.CiscoObjectContainer)
      com.cisco.jtapi.extensions.CiscoMediaTerminal
      com.cisco.jtapi.extensions.CiscoRouteTerminal


javax.telephony.TerminalConnection
   javax.telephony.callcontrol.CallControlTerminalConnection
      com.cisco.jtapi.extensions.CiscoTerminalConnection (also extends
com.cisco.jtapi.extensions.CiscoObjectContainer)


javax.telephony.TerminalObserver
   com.cisco.jtapi.extensions.CiscoTerminalObserver


com.cisco.services.tracing.TraceModule
   com.cisco.jtapi.extensions.CiscoJtapiPeer (also extends
                              com.cisco.jtapi.extensions.CiscoObjectContainer,
                                                    javax.telephony.JtapiPeer)
```

CiscoAddrActivatedEv

If an address is controlled and the restriction status changes to active, the system sends the CiscoAddrActivatedEv event to the application. Applications see this event whenever an Address or associated Terminal is in the control list. If any observers exist on the address already, applications see CiscoAddrInServiceEv. If no observers are present, applications can try to add observers, and the address will go in service.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Declaration

public interface CiscoAddrActivatedEv extends CiscoProvEv

Fields

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 22: Methods in CiscoAddrActivatedEv

Interface	Method	Description
javax.telephony.Address	getAddress()	Returns the Address which is activated.

Inherited Methods

From Interface javax.telephony.events.ProvEv

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values for more information.

Superinterfaces

javax.telephony.callcontrol.events.CallCtlCallEv, javax.telephony.callcontrol.events.CallCtlConnEv, javax.telephony.callcontrol.events.CallCtlConnOfferedEv, javax.telephony.callcontrol.events.CallCtlEv, javax.telephony.events.CallEv, javax.telephony.events.ConnEv, javax.telephony.events.Ev

Declaration

public interface CiscoCallCtlConnOfferedEv extends javax.telephony.callcontrol.events.CallCtlConnOfferedEv

Fields

None

Inherited Fields

From Interface javax.telephony.callcontrol.events.CallCtlConnOfferedEv

None

From Interface javax.telephony.callcontrol.events.CallCtlEv

CAUSE_ALTERNATE, CAUSE_BUSY, CAUSE_CALL_BACK, CAUSE_CALL_NOT_ANSWERED, CAUSE_CALL_PICKUP, CAUSE_CONFERENCE, CAUSE_DO_NOT_DISTURB, CAUSE_PARK, CAUSE_REDIRECTED, CAUSE_REORDER_TONE, CAUSE_TRANSFER, CAUSE_TRUNKS_BUSY, CAUSE_UNHOLD

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING,

META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 23: Methods in CiscoCallCtlConnOfferedEv

Interface	Method	Description
java.net.InetAddress	getCallingPartyIpAddr()	Returns the IP address of the calling party, or 0 (or null) if the IP Address is not available.

Inherited Methods

From Interface javax.telephony.callcontrol.events.CallCtlCallEv

getCalledAddress, getCallingAddress, getCallingTerminal, getLastRedirectedAddress

From Interface javax.telephony.callcontrol.events.CallCtlEv

getCallControlCause

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ConnEv

getConnection

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

None

CiscoAddrActivatedOnTerminalEv

The CiscoAddrActivatedOnTerminalEv event gets sent when a shared line gets activated or a Terminal which has shared line gets activated.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Declaration

public interface CiscoAddrActivatedOnTerminalEv extends CiscoProvEv

Fields

None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 24: Methods in CiscoAddrActivatedOnTerminalEv

Interface	Method	Description
javax.telephony.Address	getAddress()	Returns the address that is marked unrestricted on the terminal.
javax.telephony.Terminal	getTerminal()	Returns the terminal on which the address got activated (i.e. marked unrestricted).

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values for more information.

CiscoAddrAddedToTerminalEv

The system sends CiscoAddrAddedToTerminalEv when:

- A user adds a Terminal into the control list that contains a shared line, the system sends this event to the application. If a user has an address in the control list, and you add a new Terminal with the same address in control list, this event gets sent.
- An Extension Mobility (EM) user logs into a Terminal with a profile that contains a shared line, this event notifies that a new Terminal has been added to an already existing address.
- A new shared line gets added to a Terminal in a user control list, the system sends this event to the application.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Declaration

public interface CiscoAddrAddedToTerminalEv extends CiscoProvEv

Fields

Table 25: Fields in CiscoAddrAddedToTerminalEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 26: Methods in CiscoAddrAddedToTerminalEv

Interface	Method	Description
javax.telephony.Address	getAddress()	Returns the address on which the new terminal is added.
javax.telephony.Terminal	getTerminal()	Returns the terminal that gets added to the Address.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values for more information.

CiscoAddrAutoAcceptStatusChangedEv

The system sends CiscoAddrAutoAcceptStatusChangedEv to applications whenever the AutoAccept status for the Address on the Terminal changes. If an Address has multiple Terminals, this event gets sent for the Address AutoAccept status on each individual Terminal.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

javax.telephony.events.AddrEv, CiscoAddrEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoAddrAutoAcceptStatusChangedEv extends CiscoAddrEv

Fields

Table 27: Fields in CiscoAddrAutoAcceptStatusChangedEv

Interface	Field
static int	ID

Inherited Fields

From Interface javax.telephony.events.Ev

CAUSE_CALL_CANCELLED, CAUSE_DEST_NOT_OBTAINABLE, CAUSE_INCOMPATIBLE_DESTINATION, CAUSE_LOCKOUTUSE_NETWORK_CONGESTION, CAUSE_NETWORK_NOT_OBTAINABLE, CAUSE_NEW_CALL, CAUSE_NORMAL, CAUSE_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_R_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 28: Methods for CiscoAddrAutoAcceptStatusChangedEv

Interface	Method	Description
int	getAutoAcceptStatus()	Returns the AutoAccept Status of the Address on the Terminal. Returns CiscoAddress.AUTOACCEPT_OFF or CiscoAddress.AUTOACCEPT_ON
CiscoTerminal	getTerminal()	Returns the Terminal at which the AutoAccept status for this address is changing.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.AddrEv

getAddress

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See getAutoAcceptStatus and CiscoAddress.getAutoAcceptStatus(Terminal terminal).

CiscoAddrCreatedEv

The CiscoAddrCreatedEv event gets sent when an Address gets added to the provider domain.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Declaration

public interface CiscoAddrCreatedEv extends CiscoProvEv

Fields

Table 29: Fields in CiscoAddrCreatedEv

Interface	Field
ID	static final int ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 30: Methods in CiscoAddrCreatedEv

Interface	Method	Description
getAddress	javax.telephony.Address getAddress()	Returns the address which got added to the provider domain. Returns the address that is added to the provider domain.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoAddrMonitorTerminatedEv

When a monitor session is terminated, the Supervisor who had initiated the session will be notified with this event.

Interface History

Cisco Unified Communications Manager Release Number	Description
8.0(1)	New interface

Declaration

pubic interface CiscoAddrMonitorTerminatedEv extends CiscoAddrEv

Methods

Table 31: Methods in CiscoAddrMonitorTerminatedEv

Interface	Method	Description
Int	getTransactionID()	Returns the transaction ID for the session termination.
Address	getMonitorTargetAddress()	Returns the target address that was being monitored.
String	getMonitorTargetDevieName()	Returns the monitored device name.
Int	getMonitorTargetCalllegHandle()	Returns the call leg identifier for the monitored target.

Interface	Method	Description
String	getMonitorInitiatorDeviceName()	Returns the device name for the device that initiated the monitoring session.
Int	getCause()	Returns the reason that the monitoring session was terminated.

Related Documentation

CiscoAddress

The CiscoAddress interface extends the Address interface with additional Cisco Unified Communications Manager capabilities.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1, 2)	Added voice and fax message counts for the Enhanced Message Waiting Indication (MWI) feature for supported phones only.
7.1(3)	Updated for Terminal and Address Capability settings changes.
8.0(1)	Enhanced with the following: • New APIs getPickupGroup() to enable applications to get information about the Pickup Group the Address belongs to • New address type to indicate that the address represents hunt pilot. • New field that will represent a new kind of recording type, device-based recording.
9.0(1)	A new constant, SELECTIVE_RECORDING, is added. Two constants, APPLICATION_CONTROLLED_RECORDING, and DEVICE_CONTROLLED_RECORDING, are deprecated. Applications that upgrade to Release 9.0 or later releases should use the new SELECTIVE_RECORDING constant and not the deprecated APPLICATION_CONTROLLED_RECORDING and DEVICE_CONTROLLED_RECORDING constants. In Release 9.0 or later releases Unified CM and JTAPI never return the DEVICE_CONTROLLED_RECORDING constant.
10.0(1)	Enhanced with the following: • New APIs to create a persistent call and to retrieve the connection object associated to the persistent call. • a new API to create an announcement call in order to play announcements to the remote destinations.

Superinterfaces

javax.telephony.Address, CiscoObjectContainer, on page 226

Subinterfaces

CiscoIntercomAddress

Fields

Table 32: Fields in CiscoAddress

Interface	Field	Description
Static int	APPLICATION_CONTROLLED_RECORDING	Application controlled Recording is configured on the Address.
Static int	AUTO_RECORDING	Auto Recording is configured on the Address.
Static int	AUTOANSWER_OFF	AutoAnswer is off.
Static int	AUTOANSWER_UNKNOWN	AutoAnswer status is unknown.
Static int	AUTOANSWER_WITHHEADSET	AutoAnswer is allowed with a headset.
static int	AUTOANSWER_WITHSPEAKERSET	AutoAnswer is allowed with a speaker set.
public static final int	DEVICE_CONTROLLED_RECORDING	This value will be used to specify a new recording type. This type is used when the recording profile is configured on the device, and is thus “device controlled”
static int	EXTERNAL	This represents an external address with a valid name.
static int	EXTERNAL_UNKNOWN	This represents an external address with an unknown name.
static int	IN_SERVICE	The address is in service.
static int	INTERNAL	This is an internal address.
static int	MONITORING_TARGET	This represents an address with a monitoring target or agent.
static int	NO_RECORDING	Recording is off on the Address.
static int	OUT_OF_SERVICE	The address is out-of-service.
static int	RINGER_DEFAULT	Sets the ringer status to the configured value.
static int	RINGER_DISABLE	Disables the ringer for the address.

Interface	Field	Description
static int	RINGER_ENABLE	Enables the ringer for the address.
static int	SELECTIVE_RECORDING	This constant is added to replace the deprecated constants APPLICTION_CONTROLLED_RECORDING and DEVICE_CONTROLLED_RECORDING
static int	UNKNOWN	This represents an address with an unknown name.

Methods

Table 33: Methods in CiscoAddress

Interface	Method	Description
void	clearCallConnections ()	Use this interface to clear any phantom calls on the address. **Throws** javax. telephony. PrivilegeViolationException—Use this interface to clear any phantom calls on the address.
CiscoCall	createPersistentCall (Terminal terminal, String callerIDNumber, String callerIDName)	This interface creates a persistent call for this address and will return the call object for the newly created call. Note that CiscoProvider and the address must be in IN_SERVICE state, otherwise InvalidStateException will be thrown. This API cannot be invoked on external addresses. Doing so will result in MethodNotSupportedException to be thrown. If while trying to allocate a globalCallId for the persistent call and an error occurs, ResourceUnavailableException will be thrown. All other errors encountered will result in PlatformException to be thrown.
	startAnnouncement (Terminal terminal, String announcementID)	This interface creates an announcement call for this address in order to play announcements to the remote destination. It returns the call object for the newly created call. Note that CiscoProvider and the address must be in IN_SERVICE state, otherwise InvalidStateException is thrown. This API cannot be invoked on external addresses. Doing so results in MethodNotSupportedException being thrown. If while trying to allocate a globalCallId for the announcement call and an error occurs, ResourceUnavailableException ise thrown. All other errors encountered results in PlatformException being thrown.
CiscoAddressCallInfo	getAddressCallInfo (javax. telephony. Terminal terminal)	Use this interface to get information about calls that are present at the Terminal.

Interface	Method	Description
String	getAsciiLabel (Terminal term)	This method returns the ASCII label configured for this address on Terminal term. Throws InvalidStateException, MethodNotSupportedException, InValidParameterException.
int	getAutoAcceptStatus (javax. telephony. Terminal terminal)	Returns the AutoAccept status of the Address on the Terminal. **Throws** javax. telephony. PlatformException, javax. telephony. InvalidStateException, javax. telephony. MethodNotSupportedException Returns the AutoAccept status of the Address on the Terminal. It may return one of the following constants: • CiscoAddress. AUTOACCEPT_OFF • CiscoAddress. AUTOACCEPT_ON **Pre-conditions** (this. getProvider ()). getState () = = Provider. IN_SERVICE (getState () = = IN_SERVICE **Parameters** • terminal - The Terminal on which the AutoAccepts

Interface	Method	Description
int	getAutoAnswerStatus (javax. telephony. Terminal term)	This interface returns the AutoAnswer status of this Address on given Terminal. **Throws** javax. telephony. PlatformException, javax. telephony. InvalidStateException, javax. telephony. MethodNotSupportedException If return value is AUTOANSWER_OFF, that means AutoAnswer is disabled. If return value is AUTOANSWER_WITHHEADSET, that means AutoAnswer is enabled with HEADSET. If return value is AUTOANSWER_WITHSPEAKERSET, that means AutoAnswer is enabled with SPEAKERSET. If return value is AUTOANSWER_UNKNOWN, that means AutoAnswer status is UNKNOWN. **Pre-conditions** (this. getProvider ()). getState () = = Provider. IN_SERVICE (getState () = = IN_SERVICE **Parameters** • term - Terminal at which AutoAnswer is checked Returns one of the following values: • CiscoAddress. AUTOANSWER_OFF • CiscoAddress. AUTOANSWER_WITHHEADSET • CiscoAddress. AUTOANSWER_WITHSPEAKERSET • CiscoAddress. AUTOANSWER_UNKNOWN **Throws** javax. telephony. InvalidStateException - The Provider or Address is not"IN_SERVICE". javax. telephony. PlatformException - If Address is not on Terminal term javax. telephony. MethodNotSupportedException - If Address is an External Address
int	getBusyTrigger (Terminal term)	This method returns the busy trigger configured for this address on terminal term. Throws InvalidStateException, InvalidArgumentException, MethodNotSupportedException.

Interface	Method	Description
int	getButtonPosition (Terminal term)	This method returns the button position of the address on terminal term. Throws InvalidStateException, InvalidArgumentException, MethodNotSupportedException.
javax. telephony. Terminal[]	getInServiceAddrTerminals ()	Use this interface to find out which Shared Lines are in service. In Shared Lines, the same Address appears on different Terminals. Returns: Terminal[]—An array of Terminals on which the Address is in service.
int	getMaxCalls (Terminal term)	This new method returns the maximum calls configured for an address on a terminal. This method throws InvalidStateException if the associated terminal is not registered to Cisco Unified Communication Manager. It throws InvalidArgumentException if terminal does not have this address. MethodNotSupportedException is be thrown if address is not in Provider
java. lang. String	getPartition ()	It returns the partition associated with an Address.
Connection	getPersistentConnection (Terminal terminal)	This interface will return the connection object that is associated with the persistent call. It returns null if there is no persistent call. This API cannot be invoked on external addresses. Doing so will result in MethodNotSupportedException to be thrown.
CiscoPickupGroup	getPickupGroup ()	This method returns a CiscoPickupGroup object that represents the Pickup Group DN and Partition that this Address belongs to.

Interface	Method	Description
int	getRecordingConfig (javax. telephony. Terminal term)	Returns the configured recording type on this Address. **Throws** javax. telephony. PlatformException, javax. telephony. InvalidStateException, javax. telephony. MethodNotSupportedException **Returns** • int—The configured recording type on this Address. • CiscoAddess. NO_RECORDING —The call cannot be recorded. • CiscoAddress. AUTO_RECORDING—Cisco Unified Communications Manager records all answered calls to/from this address. • CiscoAddress. APPLICATION_CONTROLLED_RECORDING—Calls get recorded only when the application initiates recording. Throws javax. telephony. InvalidStateException - The Provider or Address is not"IN_SERVICE". javax. telephony. PlatformException - If Address is not on Terminal term javax. telephony. MethodNotSupportedException - If Address is an External Address
int	getRegistrationState ()	**Deprecated**. This method has been replaced by the getState () method. Returns the state of this address can be any of the following constants: • CiscoAddress. OUT_OF_SERVICE • CiscoAddress. IN_SERVICE
javax. telephony. Terminal[]	getRestrictedAddrTerminals ()	Returns the array of Terminals on which this Address is restricted. In shared lines, few lines on Terminals may be restricted. Applications cannot see any call events for restricted Addresses. If a restricted Address is involved in a call with any other controlled Terminal, the system creates a Connection for the restricted Address, but there is not any TerminalConnection for the restricted Address. Returns: Terminal[]—An array of Terminals on which this Address is restricted. If none is restricted, this method returns null.

Interface	Method	Description
int	getState ()	Returns the state of this address. The state may be any of the following constants: • CiscoAddress. OUT_OF_SERVICE • CiscoAddress. IN_SERVICE
int	getType ()	Returns the following address constants: • CiscoAddress. INTERNAL • CiscoAddress. EXTERNAL • CiscoAddress. EXTERNAL_UNKNOWN • CiscoAddress. UNKNOWN • CiscoAddress. MONITORING_TARGET • CiscoAddress. HUNT_PILOT, if address is in a CiscoHuntConnection. • CiscoAddress. HUNT_PILOT, if address represents hunt pilot.
String	getUnicodeLabel (Terminal term)	This method returns the Unicode label configured for this address on Terminal term. Throws InvalidStateException, MethodNotSupportedException, InValidParameterException.
int	getVoiceMailPilot ()	This method returns the voice mail pilot of the address. Throws InvalidStateException, MethodNotSupportedException.
boolean	isRestricted (javax. telephony. Terminal terminal)	This method returns true if this Address on Terminal is restricted. ; false if not restricted.

Interface	Method	Description
void	setAutoAcceptStatus (int autoAcceptStatus, javax. telephony. Terminal terminal)	This method lets an application enable AutoAccept for this Address on CiscoMediaTerminal and/or CiscoRouteTerminal. Addresses on CiscoTerminal other than CiscoMediaTerminal or CiscoRouteTerminal will always have AutoAccept on. If the Terminal passed in the parameter is not a CiscoMediaTerminal or CiscoRouteTerminal, this method throws an exception. For a CiscoMediaTerminal that shares an Address with CiscoTerminal, Cisco recommends enabling AutoAccept on CiscoMediaTerminal. **Throws** javax. telephony. PlatformException, javax. telephony. InvalidStateException, javax. telephony. MethodNotSupportedException **Pre-conditions** (this. getProvider ()). getState () = = Provider. IN_SERVICE (getState () = = IN_SERVICE **Post-conditions** Enables or Disables auto accept status **Parameters** • autoAcceptStatus - can be either CiscoAddress. AUTOACCEPT_OFF or CiscoAddress. AUTOACCEPT_ON. If autoAcceptStatus is AUTOACCEPT_ON, it will enable AutoAccept for Address on Terminal. If autoAcceptStatus is AUTOACCEPT_OFF, it will disable AutoAccept for Address on Terminal. • terminal - The Terminal on which AutoAccept will be enabled **Throws** javax. telephony. InvalidStateException - The Provider or Address is not "In_Service". javax. telephony. PlatformException - The Terminal does not have this Address. javax. telephony. MethodNotSupportedException - If the Terminal is not CiscoMediaTerminal or CiscoRouteTerminal.

Interface	Method	Description
void	setMessageWaiting (java. lang. String destination, boolean enable)	Specifies whether the message-waiting indicator should be activated or deactivated for the Address specified by the destination. If enable is true, message-waiting gets activated if not already activated. If enable is false, message-waiting gets deactivated if not already deactivated. **Throws** javax. telephony. MethodNotSupportedException, javax. telephony. InvalidStateException, javax. telephony. PrivilegeViolationException **Pre-conditions** (this. getProvider ()). getState () = = Provider. IN_SERVICE **Post-conditions** Enables or disables the Message Waiting Indicator depending on the enable status. **Note** This implementation currently does not enforce the post-conditions as specified in CallControlAddress as follows: this. getMessageWaiting () = = enable CallCtlAddrMessageWaitingEv gets delivered for this Address. **Parameters** • destination - DN/Address message-waiting indicator is activated/deactivated • enable - True to activate message-waiting, false to deactivate **Throws** • javax. telephony. MethodNotSupportedException—This method is not supported by the given implementation. javax. telephony. InvalidStateException **Note** The Provider is not“in service. ” javax. telephony. PrivilegeViolationException **Note** The Provider user has insufficient privileges to invoke the message-waiting indicator for this destination.

Interface	Method	Description
void	setRingerStatus (int status)	Changes the ringer status on this address. **Throws** javax. telephony. MethodNotSupportedException, javax. telephony. InvalidStateException, javax. telephony. InvalidArgumentException Accepts one of the following constants: • CiscoAddress. RINGER_DEFAULT • CiscoAddress. RINGER_DISABLE • CiscoAddress. RINGER_ENABLE

Interface	Method	Description
void	setMessageSummary (boolean enable, boolean voiceCounts, int totalNewVoiceMsgs, int totalOldVoiceMsgs, boolean highPriorityVoiceCounts, int newHighPriorityVoiceMsgs, int oldHighPriorityVoiceMsgs, boolean faxCounts, int totalNewFaxMsgs, int totalOldFaxMsgs, boolean highPriorityFaxCounts, int newHighPriorityFaxMsgs, int oldHighPriorityFaxMsgs)	

Interface	Method	Description
		Use this interface to set the message-waiting indicator along with voice/fax message waiting counts If enable is true, message-waiting gets activated if not already activated. If enable is false, message-waiting gets deactivated if not already deactivated. **Pre-conditions** (this. getProvider ()). getState () = = Provider. IN_SERVICE **Post-conditions** Enables or disables the Message Waiting Indicator and sets message waiting counts. **Parameters** • enable - True to activate message-waiting, false to deactivate • voiceCounts - indicates if voice message counts are provided • totalNewVoiceMsgs - specifies the total number of new voice messages waiting • totalOldVoiceMsgs - specifies the total number of old voice messages waiting • highPriorityVoiceCounts - indicates if high priority voice message counts are provided • newHighPriorityVoiceMsgs - specifies the number of new high priority voice messages waiting • oldHighPriorityVoiceMsgs - specifies the number of old high priority voice messages waiting • faxCounts - indicates if fax message counts are provided • totalNewFaxMsgs - specifies the total number of new fax messages waiting • totalOldFaxMsgs - specifies the total number of old fax messages waiting • highPriorityFaxCounts - indicates if high priority fax message counts are provided • newHighPriorityFaxMsgs - specifies the number of new high priority fax messages waiting • oldHighPriorityFaxMsgs - specifies the number of old high priority fax messages waiting **Throws** javax. telephony. MethodNotSupportedException - This method is not supported by the given implementation. javax. telephony. InvalidStateException - The Provider is not "in service. "

Interface	Method	Description
		javax. telephony. PrivilegeViolationException - The Provider user has insufficient privileges to set the message-waiting indicator or message counts for this destination.
void	setMessageSummary (java. lang. String destination, boolean enable, boolean voiceCounts, int totalNewVoiceMsgs, int totalOldVoiceMsgs, boolean highPriorityVoiceCounts, int newHighPriorityVoiceMsgs, int oldHighPriorityVoiceMsgs, boolean faxCounts, int totalNewFaxMsgs, int totalOldFaxMsgs, boolean highPriorityFaxCounts, int newHighPriorityFaxMsgs, int oldHighPriorityFaxMsgs)	Use this interface to set the message-waiting indicator along with voice/fax message waiting counts for the Address specified by the destination **Pre-conditions** (this. getProvider ()). getState () = = Provider. IN_SERVICE **Post-conditions** Enables or disables the Message Waiting Indicator and sets message waiting counts. **Parameters** • destination - DN/Address whose message-waiting indicator should be activated/deactivated • enable - True to activate message-waiting, false to deactivate • voiceCounts - indicates if voice message counts are provided • totalNewVoiceMsgs - specifies the total number of new voice messages waiting • totalOldVoiceMsgs - specifies the total number of old voice messages waiting • highPriorityVoiceCounts - indicates if high priority voice message counts are provided • newHighPriorityVoiceMsgs - specifies the number of new high priority voice messages waiting • oldHighPriorityVoiceMsgs - specifies the number of old high priority voice messages waiting • faxCounts - indicates if fax message counts are provided • totalNewFaxMsgs - specifies the total number of new fax messages waiting • totalOldFaxMsgs - specifies the total number of old fax messages waiting • highPriorityFaxCounts - indicates if high priority fax message counts are provided • newHighPriorityFaxMsgs - specifies the number of new high priority fax messages waiting • oldHighPriorityFaxMsgs - specifies the number of old high priority fax messages waiting

Inherited Methods

From Interface javax.telephony.Address

addCallObserver, addObserver, getAddressCapabilities, getCallObservers, getCapabilities, getConnections, getName, getObservers, getProvider, getTerminals, removeCallObserver, removeObserver

From Interface com.cisco.jtapi.extensions.CiscoObjectContainer

getObject, setObject

Parameters

- Terminal terminal: The terminal object you want to create the persistent call for.
- String callerIDNumber: The number you wish to show up on the remote destination's Caller ID.
- String callerIDName: The name you wish to show up on the remote destination's Caller ID.

Related Documentation

See Constant Field Values for more information.

CiscoAddressObserver

Applications implement this interface to receive CiscoAddrEv events such as CiscoAddrInServiceEv 0 CiscoAddrOutOfServiceEv when observing Addresses via the Address.addObserver method.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

javax.telephony.AddressObserver

Declaration

public interface CiscoAddressObserver extends javax.telephony.AddressObserver

Fields

None

Methods

None

Inherited Methods

From Interface javax.telephony.AddressObserver

addressChangedEvent

Related Documentation

See CiscoAddrInServiceEv, CiscoAddrOutOfServiceEv for more information.

CiscoAddrEv

The CiscoAddrEv interface extends the JTAPI core javax.telephony.events.AddrEv interface and serves as the base interface for all Cisco extended JTAPI Address events. Every Address related event in this package extends this interface, directly or indirectly.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

javax.telephony.events.AddrEv, CiscoEv, javax.telephony.events.Ev

Subinterfaces

CiscoAddrAutoAcceptStatusChangedEv, CiscoAddrInServiceEv, CiscoAddrIntercomInfoChangedEv, CiscoAddrIntercomInfoRestorationFailedEv, CiscoAddrOutOfServiceEv, CiscoAddrRecordingConfigChangedEv

Declaration

public interface CiscoAddrEv extends CiscoEv, javax.telephony.events.AddrEv

Fields

None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.AddrEv

getAddress

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See javax.telephony.events.AddrEv for more information.

CiscoAddrEvFilter

CiscoAddrEvFilter provided for applications to set filters for address events. The application can use the following APIs to enable/disable the filters to receive the event notifications on address or to check the value

set of the filter. Application can enable the filter, if it wishes to receive the new event (CiscoAddrParkStatusEv), for the rest of the events the filter values are true by default.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Added this event for Park Monitoring and Assisted DPark Support feature.
7.1.(3)	Interface is enhanced to allow application set filter on address to enable and disable CiscoAddrVoiceMailPilotChangedEv.
8.0(1)	Enhanced with the following: • getCiscoAddrMonitoringTerminatedEvFilter() • setCiscoAddrMonitoringTerminatedEvFilter() By default the filter will be set to 'true' and CiscoMonitoringTerminatedEv will be delivered. To stop receiving this event applications need to set the filter to false.

Fields

None

Methods

Table 34: Methods in CiscoAddrEvFilter

Interface	Method	Description
boolean	getCiscoAddrParkStatusEvFilter ()	Application can invoke this API to know status of the filter for CiscoAddrParkStatusEv. Default value returned is false.
boolean	getCiscoAddrIntercomInfoChangedEvFilter ()	Application can invoke this API to know the stutus of the filter for CiscoAddrIntercomInfoChangedEv. Default value is true.
boolean	getCiscoAddrIntercomInfoRestorationFailedEvFilter ()	Application can invoke this API to know the status of the filter for CiscoAddrIntercomInfoRestorationFailedEv. Default value is true.
boolean	getCiscoAddrMonitorTerminatedEvFilter ()	Application can invoke this API to know the status of the filter for CiscoAddrMonitorTerminatedEv. Default value is true.

Interface	Method	Description
boolean	getCiscoAddrRecordingConfigChangedEvFilter ()	Application can invoke this API to get the status of the filter for the CiscoAddrRecordingConfigChangedEv. The default value is true.
boolean	getCiscoAddrVoiceMailPilotChangedEvFilter ()	This method returns true if voice mail pilot changed event filter is turned on else false.
void	setCiscoAddrIntercomInfoChangedEvFilter (boolean filter value)	Application can invoke this API to set the status of the filter for CiscoAddrIntercomInfoChangedEv.
void	setCiscoAddrIntercomInfoRestorationFailedEvFilter (boolean filter value)	Application can invoke this API to set the status of the filter for CiscoAddrIntercomInfoRestorationFailedEv.
void	setCiscoAddrMonitorTerminatedEvFilter (Boolean filterValue)	**Parameter** Boolean
Void	setCiscoAddrParkStatusEvFilter (Boolean filterValue)	Application can invoke this API to set the status of the filter for CiscoAddrParkStatusEv.
void	setCiscoAddrRecordingConfigChangedEvFilter (boolean filter value)	Application can invoke this API to set the value of the filter for CiscoAddrRecordingConfigChangedEv.
void	setCiscoAddrVoiceMailPilotChangedEvFilter (boolean filterValue)	This method enables or disables the address voice mail changed event. When this filter is turned on CiscoAddrVoiceMailPilotChangedEv is delivered to address observer when voice mail configuration is changed.

Sample Code

```
CiscoAddress caddr  =  (CiscoAddress) provider.getAddress("2000");
If ( caddr ! =  null ){
   CiscoAddrEvFilter filter  =  caddr.getFilter();
   filter.setCiscoAddrVoiceMailPilotChangedEvFilter(true);
   caddr.addObserver(myAddrObserver);
}

try {
    JtapiPeer peer  =  JtapiPeerFactory.getJtapiPeer ( null );
    MyProviderObserver providerObserver  =  new MyProviderObserver ();
    provider  =  peer.getProvider ( ipaddress;login = useid;passwd = password );
    if ( provider ! =  null ) {
       provider.addObserver ( providerObserver );
       provInService.waitTrue();
       CiscoAddrEvFilter filter;
       CiscoAddress addr  =  provider.getAddress(S1,  S1p);
       filter.setCiscoAddrMonitoringTerminatedEvFilter(false);
       addr.setFilter(filter);
       System.out.println(" Current filter value is : "+
       addr.getFilter().getCiscoAddrMonitoringTerminatedEvFilter());
    }
```

Inherited Methods

None

Parameters

The set methods take a Boolean value as the parameter.

Value Range

The get methods return a Boolean value (true or false).

Related Documentation

See Constant Field Values.

CiscoAddrInServiceEv

The CiscoAddrInServiceEv indicates that the Address is now IN_SERVICE. With Shared Lines (where the same Address appears on different Terminals), applications may receive multiple CiscoAddressInService events for all the Terminals. Applications can use this interface to find out the Terminal on which the Address (or Shared Line) is going IN_SERVICE.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

javax.telephony.events.AddrEv, CiscoAddrEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoAddrInServiceEv extends CiscoAddrEv

Fields

Table 35: Fields in CiscoAddrInService

Interface	Field
Static int	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 36: Methods in CiscoAddrInService

Interface	Method	Description
getTerminal	CiscoTerminal getTerminal()	Returns the Terminal at which this Address is going IN_SERVICE.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.AddrEv

getAddress

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Related Documentation, on page 41.getInServiceAddrTerminals() and Constant Field Values for more information.

CiscoAddrIntercomInfoChangedEv

The system sends the CiscoAddrIntercomInfoChangedEv event to the application whenever the target DN or intercom target label changes for a CiscoIntercomAddress. The system provides this event to all of the application observers that have been added to the CiscoIntercomAddress.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

javax.telephony.events.AddrEv, CiscoAddrEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoAddrIntercomInfoChangedEv extends CiscoAddrEv

Fields

Table 37: Fields in CiscoAddrIntercomInfoChangedEv

Interface	Field
Static Int	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN,

META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 38: Methods in CiscoAddrIntercomInfoChangedEv

Interface	Method	Description
getIntercomAddress	getIntercomAddress()	Returns the intercom address for which the information changed.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.AddrEv

getAddress

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See CiscoAddrEv and Constant Field Values for more information.

CiscoAddrIntercomInfoRestorationFailedEv

The system sends the CiscoAddrIntercomInfoRestorationFailedEv event to the application when JTAPI cannot restore the application set intercom target DN or the intercom target label for the CiscoIntercomAddress during failover or fallback. The system provides this event on the application observer for the application that set the intercom target DN or the intercom target label.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

javax.telephony.events.AddrEv, CiscoAddrEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoAddrIntercomInfoRestorationFailedEv extends CiscoAddrEv

Fields

Table 39: Fields in CiscoAddrIntercomInfoRestorationFailedEv

Interface	Field
Static int	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 40: Methods in CiscoAddrIntercomInfoRestorationFailedEv

Interface	Method	Description
CiscoIntercomAddress	getIntercomAddress()	This interface returns the Cisco IntercomAddress for which intercom information restoration failed.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.AddrEv

getAddress

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values and CiscoAddrEv for additional information.

CiscoAddrPickupGroupChangedEv

CiscoAddrPickupGroupChangedEv is a new interface being added with Call Pickup feature development. This event is fired whenever a pickup group's information changes, and the line info gets updated. The line info will only be updated when the line is updated with the "apply config" button in the CUCM.

Interface History

Cisco Unified Communications Manager Release Number	Description
8.0(1)	New interface

Declaration

public interface CiscoAddrPickupGroupChangedEv extends CiscoProvEv

Methods

Table 41: Methods in CiscoAddrPickupGroupChangedEv

Interface	Method	Description
CiscoPickupGroup	getOldPickupGroup()	This method returns the old Pickup Group information for this event.
CiscoPickupGroup	getNewPickupGroup()	This method returns the new Pickup Group information for this event, what the pickup group has changed to.

New Error Code

CTIERR_PICKUPGROUP_CHANGED

CTIERR_PICKUPGROUP_DELETED

CiscoAddrOutOfServiceEv

The CiscoAddrOutOfServiceEv event notifies applications that an Address has gone OUT_OF_SERVICE. With Shared Lines(where the same Address appears on different Terminals), applications may receive multiple CiscoAddrOutOfServiceEv events for all the Terminals. Applications can use this interface to find out the Terminal on which the Address(or Shared Line) is going OUT_OF_SERVICE.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

javax.telephony.events.AddrEv, CiscoAddrEv, CiscoEv, CiscoOutOfServiceEv, javax.telephony.events.Ev

Declaration

public interface CiscoAddrOutOfServiceEv extends CiscoAddrEv, CiscoOutOfServiceEv

Fields

Table 42: Fields in CiscoAddrOutOfServiceEv

Interface	Field
Static int	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface com.cisco.jtapi.extensions.CiscoOutOfServiceEv

CAUSE_CALLMANAGER_FAILURE, CAUSE_CTIMANAGER_FAILURE, CAUSE_DEVICE_FAILURE, CAUSE_DEVICE_RESTRICTED, CAUSE_DEVICE_UNREGISTERED, CAUSE_LINE_RESTRICTED, CAUSE_NOCALLMANAGER_AVAILABLE, CAUSE_REHOME_TO_HIGHER_PRIORITY_CM, CAUSE_REHOMING_FAILURE

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 43: Methods in CiscoAddrOutOfServiceEv

Interface	Method	Description
CiscoTerminal	getTerminal()	Returns the Terminal at which this Address is going OUT_OF_SERVICE.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.AddrEv

getAddress

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values and Related Documentation, on page 41.getInServiceAddrTerminals() for more information.

CiscoAddrParkStatusEv

When parking a call using the Cisco Unified IP Phone, JTAPI reports park states by using this event. It is provided to all the applications, which have address observers added on the address, which has invoked park. This event gets delivered only when park gets invoked from Cisco Unified IP Phones.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Added interface for Park Monitoring and Assisted DPark feature.

Declaration

public interface CiscoAddrParkStatusEv extends CiscoAddrEv

Fields

Table 44: Fields in CiscoAddrParkStatusEv

Interface	Field	Description
static int	PARKED	Park status when the call is parked.
static int	REMINDER	Park status when the park monitoring reversion timer expires.
static int	RETRIEVED	Park status when the parked call is retrieved by the parker or a third party.
static int	FORWARDED	Park status when the parked call is forwarded when the park monitoring forward- no-retrieve timer expires.
static int	ABANDONED	Park status when the parked call is disconnected.

Inherited Fields

None

Methods

Table 45: Methods in CiscoAddrParkStatusEv

Interface	Method	Description
int	getParkState()	Returns the current park state of the parked call.
int	getTransactionID()	Returns an id which is unique for a particular parked call. Transaction ID would remain the same in the different park states for the same parked call.
CiscoCallID	getCiscoCallID()	Returns CiscoCallID.
String	getParkDN()	Returns the DN where call is parked.
String	getParkDNPartition()	Returns the partition of the Park DN.
String	getParkedParty()	Returns the DN of the parked party.
String	getParkedPartyPartition()	Returns the partition of the Parked party.
Terminal	getTerminal()	Returns the terminal on whose address this event is delivered.

Value Ranges

The following are values of fields:

- PARKED: 2
- REMINDER: 3
- RETRIEVED: 4
- ABANDONED: 5
- FORWARDED: 6

Related Documentation

See Constant Field Values for more information.

CiscoAddrRecordingConfigChangedEv

The system delivers the CiscoAddrRecordingConfigChangedEv event to the Address Observer if the recording setting on the Address changes.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

javax.telephony.events.AddrEv, CiscoAddrEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoAddrRecordingConfigChangedEv extends CiscoAddrEv

Fields

Table 46: Fields in CiscoAddrRecordingConfigChangedEv

Interface	Field
static int	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING,

META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 47: Methods in CiscoAddrRecordingConfigChangedEv

Interface	Method	Description
Int	getRecordingConfig()	Returns the new recording configuration on this Address. The value is one of the following: • CiscoAddress.NO_RECORDING • CiscoAddress.AUTO_RECORDING • CiscoAddress.APPLICATION_CONTROLLED_RECORDING
javax.telephony.Terminal	getTerminal()	Returns the Terminal on which the recording configuration changed.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.AddrEv

getAddress

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values and CiscoAddrEv for more information.

CiscoAddrRemovedEv

JTAPI sends the CiscoAddrRemovedEv event when an Address gets removed from the Provider domain.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Declaration

public interface CiscoAddrRemovedEv extends CiscoProvEv

Fields

Table 48: Fields in CiscoAddrRemovedEv

Interface	Field
static int	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 49: Methods in CiscoAddrRemovedEv

Field	Method	Description
javax.telephony.Address	getAddress()	Returns the Address that is removed from provider domain and the address which is removed from the user control list.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values for more information.

CiscoAddrRemovedFromTerminalEv

The system sends the CiscoAddrRemovedFromTerminalEv event under the following conditions:

- When an Administrator removes a Terminal from the user control list that contains a Shared Line.
- When an Extension Mobility (EM) user logs out from a Terminal with a profile that contains a Shared Line, this event notifies that one of the Terminals got removed from an existing Address.
- When a Shared Line is removed from a Terminal that is in a user control list.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Declaration

public interface CiscoAddrRemovedFromTerminalEv extends CiscoProvEv

Fields

Table 50: Fields in CiscoAddrRemovedFromTerminalEv

Interface	Field
Static int	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 51: Methods in CiscoAddrRemovedFromTerminalEv

Interface	Method	Description
javax.telephony.Address	getAddress()	Returns the Address from which the Terminal got removed.
javax.telephony.Terminal	getTerminal()	Returns the Terminal that got removed from the Address.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See also Constant Field Values for more information.

CiscoAddrRestrictedEv

If an Address is observed and the restriction status is changed to restricted, the system sends this event to the application.

Applications will see this event whenever an Address or an associated Terminal is Restricted from Cisco Unified Communications Manager Administration. For restricted lines, the Address will go OUT_OF_SERVICE and will not come back IN_SERVICE until it is activated again. If an Address is restricted, addCallObserver and addObserver will throw an exception.

For shared lines, if a few shared lines are restricted, and others are not, the system does not throw an exception but the restricted shared lines will not receive any events. If all shared lines are restricted, an exception is thrown when application try adding observers. If an Address gets restricted after observers are added, applications will see CiscoAddrOutOfServiceEv, and when the Address is activated, the Address will go IN_SERVICE.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, CiscoRestrictedEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Declaration

public interface CiscoAddrRestrictedEv extends CiscoRestrictedEv

Fields

None

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoRestrictedEv

CAUSE_UNKNOWN, CAUSE_UNSUPPORTED_DEVICE_CONFIGURATION, CAUSE_UNSUPPORTED_PROTOCOL, CAUSE_USER_RESTRICTED

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 52: Methods in CiscoAddrRestrictedEv

Interface	Method	Description
javax.telephony.Address	getAddress()	Returns the Address which is changed to Restricted on Cisco Unified Communications Manager.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values for more information.

CiscoAddrRestrictedOnTerminalEv

If the user has Shared lines in the control list, and one of those lines is marked restricted on Cisco Unified Communications Manager, the system sends this event.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, CiscoRestrictedEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Declaration

public interface CiscoAddrRestrictedOnTerminalEv extends CiscoRestrictedEv

Fields

Table 53: Fields in CiscoAddrRestrictedOnTerminalEv

Interface	Field
Static int	ID

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoRestrictedEv

CAUSE_UNKNOWN, CAUSE_UNSUPPORTED_DEVICE_CONFIGURATION, CAUSE_UNSUPPORTED_PROTOCOL, CAUSE_USER_RESTRICTED

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS,

META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 54: Methods in CiscoAddrRestricedOnTerminalEv

Interface	Method	Description
javax.telephony.Address	getAddress()	Returns the Address that is restricted.
javax.telephony.Terminal	getTerminal()	Returns the Terminal on which the Address is restricted.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values for more information.

CiscoAddrVoiceMailPilotChangedEv

This event indicates that the voice mail pilot configuration on address is changed and is delivered to address observer. Application needs to enable the corresponding filter in CiscoAddrEvFilter to receive this event.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(3)	New interface.

Sample Code:

```
class myAddrObserver  extends CiscoAddressObserver {
  public synchronized void addressChangedEvent ( AddrEv [] eventList ) {
    if ( eventList[i] instanceof CiscoAddrVoiceMailPilotChangedEv){
       CiscoAddrVoiceMailPilotChangedEv ev  =  (CiscoAddrVoiceMailPilotChangedEv)

                                      eventList[i];
       Address cAddr  =  ev.getAddress();
       String newVoiceMailPilot  =  ev.getVoiceMailPilot();
       System.out.println(" New voice mail pilot for " +
         ev.getAddress()  +   " is " +  newVoiceMailPilot );
    }
  }
}
```

Superinterfaces

NA

Declaration

NA

Fields

Table 55: Fields in CiscoAddrVoiceMailPilotChangedEv

Interface	Field

Inherited Fields

Methods

Table 56: Methods in CiscoAddrVoiceMailPilotChangedEv

Interface	Method	Description
String	getVoiceMailPilot()	This method returns the new voice mail pilot of the address.

Inherited Methods

Related Documentation

See Constant Field Values for more information.

CiscoAnnouncementStartedEv

CiscoAnnouncementStartedEv is a new JTAPI event that is delivered to applications as a Call Event. This new event is delivered to call observers added by applications to notify when a play announcement starts.

Interface History

Cisco Unified Communications Manager Release Number	Description
10.01	Created history table to track changes

Declaration

Public interface CiscoAnnouncementStartedEv extends CiscoCallEv.

Methods

Interface	Method	Description
String	getAnnouncementID ()	This interface returns the name of the announcement identifier.

CiscoAnnouncementEndedEv

CiscoAnnouncementEndedEv is a new JTAPI event that is delivered to applications as a Call Event. This new event is delivered to call observers added by applications to notify when play announcement ends.

Interface History

Cisco Unified Communications Manager Release Number	Description
10.01	Created history table to track changes

Declaration

Public interface CiscoAnnouncementEndedEv extends CiscoCallEv.

Methods

Interface	Method	Description
boolean	getSuccess()	This interface returns whether or not the play announcement was successful. Returns true if there are no errors with the play announcement, or returns false indicating error.
int	getErrorCode()	This interface returns the error code indicating the cause of the failure/error with play announcment. This maps to one of the values defined in CiscoJtapiException.
String	getErrorDescription()	This interface returns the string corresponding to what the error code maps to.

CiscoAnnouncementErrorEv

CiscoAnnouncementErrorEv is a new JTAPI event that is delivered to applications as a Call Event. This new event is delivered to call observers added by applications to notify when an error occurs during play announcement.

Interface History

Cisco Unified Communications Manager Release Number	Description
10.01	Created history table to track changes

Declaration

Public interface CiscoAnnouncementErrorEv extends CiscoCallEv.

Methods

Interface	Method	Description
Int	getErrorCode()	This interface returns the error code indicating the cause for the failure/error with play announcment. This maps to one of the values defined in CiscoJtapiException.
String	getErrorDescription()	This interface returns the string corresponding to what the error code maps to.

CiscoBaseMediaTerminal

The CiscoBaseMediaTerminal interface extends the CiscoTerminal interface.

Interface History

Cisco Unified Communications Manager Release Number	Description
8.5(1)	New interface.

Declaration

public interface CiscoBaseMediaTerminal extends CiscoTerminal

Superinterfaces

NA

Fields

Table 57: Fields in CiscoBaseMediaTerminal

Interface	Field
Final static int	NO_MEDIA_REGISTRATION
Final static int	DYNAMIC_MEDIA_REGISTRATION
Final static int	DYNAMIC_MEDIA_REGISTRATION_FOR_GET_PORT_SUPPORT
Final static int	STATIC_MEDIA_REGISTRATION
Final static int	STATIC_MEDIA_REGISTRATION_FOR_GET_PORT SUPPORT

Inherited Fields

NA

Methods

Table 58: Methods in CiscoBaseMediaTerminal

Interface	Method
int	getRegistrationType()
void	register(java.net.InetAddress address, int port, CiscoMediaCapability[] capabilities, int registrationType), int[] algorithmIDs, java.net.InetAddress address_v6, int activeAddressingMode) throws CiscoRegistrationException, PrivilegeViolationException;

Inherited Methods

NA

Parameters

- register()
- Java.net.InteAddress address
- int port
- CiscoMediaCapability[] capabilities
- int[] algorithmIDs
- Java.net.InteAddress address_v6
- int activeAddressingMode
- int registrationType

Data Types

- Java.net.InteAddress address
- int port
- CiscoMediaCapability[] capabilities
- int[] algorithmIDs
- Java.net.InteAddress address v6
- int activeAddressingMode
- int registrationType

Range of Values

- activeAddressingMode:
  - CiscoTerminal.IP_ADDRESSING_MODE_IPv4 or
  - CiscoTerminal.IP_ADDRESSING_MODE_IPv6 or
  - CiscoTerminal.IP_ADDRESSING_MODE_IPv4_v6
- registrationType:
  - CiscoTerminal.NO_MEDIA_REGISTATION (applicable only for route points)
  - CiscoTerminal.DYNAMIC_MEDIA_REGISTRATION
  - CiscoTerminal.DYNAMIC_MEDIA_REGISTRATION_FOR_GET_PORT_SUPPORT
  - CiscoTerminal.STATIC_MEDIA_REGISTRATION

• CiscoTerminal.STATIC_MEIDA_TERMINAL_FOR_GET_PORT_SUPPORT

CiscoCall

The CiscoCall interface extends the CallControlCall interface with additional Cisco Unified Communications Manager specific capabilities.

In Cisco Unified Communications Manager, every Call object comprises a set of call legs that share a common identifier: the global call handle. Connection objects represent call legs in JTAPI, and the Call object that relates a set of connections contains the global call handle that the underlying call legs share.

The global call handle within a CiscoCall is accessible by using CallManagerID and CallID properties. Taken together, the CallManagerID and CallID form the global call handle that Cisco Unified Communications Manager maintains. Consider this pair of properties as guaranteed to be unique among all ACTIVE Call objects, but when an ACTIVE call becomes INACTIVE, its CallManagerID and CallID may be reused to identify a newly created Call object. Therefore, an INACTIVE Call can have identical CallManagerID and CallID properties to those of a currently ACTIVE Call object.

Interface History

Cisco Unified Communications Manager Release Number	Description
10.0(1)	Two new APIs: CiscoMultiMediaCapabilityInfogetCallingTerminalMultiMediaCapabilityInfo() Returns the calling party terminal multimedia capability. CiscoMultiMediaCapabilityInfogetCalledTerminalMultiMediaCapabilityInfo() Returns the called party terminal multimedia capability. Three new constants: CALL_RECORDING_MEDIA_FORKING_DEVICE_TYPE_NONE CALL_RECORDING_MEDIA_FORKING_DEVICE_TYPE_PHONE CALL_RECORDING_MEDIA_FORKING_DEVICE_TYPE_G
9.0(1)	Five new constants: CALL_RECORDING_TYPE_NONE, CALL_RECORDING_TYPE_AUTOMATIC, CALL_RECORDING_TYPE_APPLICATION_INITIATED_SILENT, CALL_RECORDING_TYPE_USER_INITIATED_FROM_APPLICATION, and CALL_RECORDING_TYPE_USER_INITIATED_FROM_DEVICE, are added.
8.0(1)	Enhanced with the following: New methods that allow applications to get the Terminals associated with the current calling and current called parties on the call. New API to indicate whether the call is created due to CallFwdAll key press or not. Three new constants, CFWD_ALL_NONE, CFWD_ALL_SET, and CFWD_ALL_CLEAR, have been introduced for CiscoCall interface.

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Added new method called isConference() for Drop Any Party feature.

Superinterfaces

javax.telephony.Call, javax.telephony.callcontrol.CallControlCall, CiscoObjectContainer

Subinterfaces

CiscoConsultCall

Declaration

public interface CiscoCall extends javax.telephony.callcontrol.CallControlCall, CiscoObjectContainer

Fields

Table 59: Fields in CiscoCall

Interface	Field	Description
static int	CALL_RECORDING_TYPE_APPLICATION_INITIATED_SILENT	This constant is used when silent is the recording invocation type. Silent recording is the default value in releases prior to Release 9.0.
static int	CALL_RECORDING_TYPE_AUTOMATIC	This constant is used when recording is invoked automatically by Unified CM, as a result of the line configuration.
static int	CALL_RECORDING_TYPE_NONE	This constant is used when a call is not recorded.
static int	CALL_RECORDING_TYPE_USER_INITIATED_FROM_APPLICATION	This constant is used when user is the recording invocation type, and the request was invoked by an application.
static int	CALL_RECORDING_TYPE_USER_INITIATED_FROM_DEVICE	This constant is used when recording was invoked on the Cisco IP device.
Static int	CALLSECURITY_AUTHENTICATED	Call security status is authenticated.
Static int	CALLSECURITY_ENCRYPTED	Call security status is encrypted.
Static int	CALLSECURITY_NOTAUTHENTICATED	Call security status is not authenticated.

Interface	Field	Description
Static int	CALLSECURITY_UNKNOWN	Call security status is unknown.
public static final int	CFWD_ALL_NONE	When call is not created due to CallFwdAll soft key press. Value is 0.
public static final int	CFWD_ALL_SET	When call is created due to CallFwdAll key press to set CFA. Value is 64.
public static final int	CFWD_ALL_CLEAR	When call is created to CallFwdAll key press to clear/cancel CFA. Value is 128.
Static int	FEATUREPRIORITY_EMERGENCY	Feature priority is emergency
Static int	FEATUREPRIORITY_NORMAL	Feature priority is normal
Static int	FEATUREPRIORITY_URGENT	Feature priority is urgent
int	getCFwdAllKeyPressIndicator()	This interface indicates if call is created due to callFWDAll Key press. It returns one of the following: • CiscoCall.CFWD_ALL_NONE • CiscoCall.CFWD_ALL_SET • CiscoCall.CFWD_ALL_CLEAR
Static int	PLAYTONE_BOTHLOCALANDREMOTE	A tone plays to both the caller and the monitor target (agent) when this option gets used.
Static int	PLAYTONE_LOCALONLY	A tone plays only to the monitor target (agent) when this option gets used.
Static int	PLAYTONE_NOLOCAL_OR_REMOTE	When this option is used no tone plays to the monitor target (agent) or the caller.
Static int	PLAYTONE_REMOTEONLY	A tone plays only to the caller when this option gets used.
Static int	SILENT_MONITOR	This option indicates that silent monitor is requested.
static final int	CALL_RECORDING_MEDIA_FORKING_DEVICE_TYPE_NONE	This option indicates that there is no Media Forking Device for recording on this call. Range of value = 0

Interface	Field	Description
static final int	CALL_RECORDING_MEDIA_FORKING_DEVICE_TYPE_PHONE	This option indicates that the Media Forking Device type for recording on this call is Phone (BIB Recording). Range of value = 1
static final int	CALL_RECORDING_MEDIA_FORKING_DEVICE_TYPE_GW	This option indicates that the Media Forking Device type for recording on this call is Gateway (GW Recording). Range of value = 2

Inherited Fields

From Interface javax.telephony.Call

ACTIVE, IDLE, INVALID

Sample Code

```
public class MyCallObserver implements implements CallObserver,
           CallControlCallObserver,  MediaCallObserver {
  public void callChangedEvent (CallEv[] evlist) {
    for(int i = 0; evlist ! =  null && i < evlist.length; i++){
…
…
      If (evlisth[i] instance of TermConnActiveEv){
         CiscoCall thisCall  =  (CiscoCall) ((TermConnActiveEv)
evlist[i]).getCall();
         int cfaStatus  =  thisCall.getCFWDAllKeyPressIndicator();
         If (cfaStatus  =  =  CiscoCall.CFWD_ALL_SET ||
             cfaStatus  =  =  CiscoCall.CFWD_ALL_CLEAR){
          System.out.println("Call is created due to CallFwdAll soft key press");

         }else {
            System.out.println("Call is NOT created due to CallFwdAll soft key
press");
      }
    }
  }
        …
        …
}
```

Methods

Table 60: Methods in CiscoCall

Interface	Method	Description
Void	conference (javax. telephony. Call[]otherCalls)	This interface conferences multiple calls together, resulting in the union of the participants of all the calls being placed on a single call.
java. lang. boolean	isConference ()	Returns True if it is a conference call, false or otherwise.
javax. telephony. Connection[]	connect (javax. telephony. Terminal origterm, javax. telephony. Addressorigaddr java. lang. String. dialedDigits int featurePriority)	This method overloads Call. connect (). It takes a new parameter featurePriority. The featurePriority parameter may be: CiscoCall. FEATUREPRIORITY_NORMAL CiscoCall. FEATUREPRIORITY_URGENT CiscoCall. FEATUREPRIORITY_EMERGENCY Throws: javax. telephony. ResourceUnavailableException, javax. telephony. PrivilegeViolationException, javax. telephony. InvalidPartyException, javax. telephony. InvalidArgumentException, javax. telephony. InvalidStateException, javax. telephony. MethodNotSupportedException
boolean	getCalledAddressPI ()	Returns the Presentation Indicator (PI) that is associated with getCalledAddressPI.
CiscoPartyInfo	getCalledPartyInfo ()	Returns the PartyInfo of the called party of the call.
CiscoCallID	getCallID ()	CallID is a unique identifier among all ACTIVE calls with the same CallManagerID.
boolean	getCallingAddressPI ()	Returns the Presentation Indicator (PI) that is associated with getCallingAddressPI.
int	getCallSecurityStatus ()	This interface returns the SecurityStatus of the Call.
CiscoConference Chain	getConferenceChain ()	This interface returns a CiscoConferenceChain object if this Call is a chained conference Call.
javax. telephony. Address	getCurrentCalledAddress ()	Returns the current called Address for the call.
boolean	getCurrentCalledAddressPI ()	Returns the Presentation Indicator (PI) that is associated with CurrentCalledAddress.
boolean	getCurrentCalledDisplayNamePI ()	Returns the Presentation Indicator (PI) that is associated with getCurredCalledDisplayNamePI.

Interface	Method	Description
java. lang. String	getCurrentCalledPartyDisplayName ()	This interface returns the display name of the called party in the call.
CiscoPartyInfo	getCurrentCalledPartyInfo ()	Returns the PartyInfo of the current called party of the call.
java. lang. String	getCurrentCalledPartyUnicodeDisplayName ()	Returns the Unicode display name of the called party in the call.
int	getCurrentCalledPartyUnicodeDisplayNamelocale ()	Returns the locale of the current called party Unicode display name.
javax. telephony. Address	getCurrentCallingAddress ()	Returns the current calling Address for the call.
boolean	getCurrentCallingAddressPI ()	Returns the Presentation Indicator (PI) that is associated with getCurrentCallingAddressPI.
boolean	getCurrentCallingDisplayNamePI ()	Returns the Presentation Indicator (PI) that is associated with getCurrentCalledDisplayNamePI.
java. lang. String	getCurrentCallingPartyDisplayName ()	This interface returns the display name of the calling party.
CiscoPartyInfo	getCurrentCallingPartyInfo ()	Returns the PartyInfo of the current calling party of the call.
java. lang. String	getCurrentCallingPartyUnicodeDisplayName ()	Returns the Unicode display name of the calling party in the call.
int	getCurrentCallingPartyUnicodeDisplayNamelocale ()	Returns the locale of the current called party Unicode display name.
Terminal	getCurrentCallingTerminal ()	This method returns a Terminal object that represents the terminal of the calling party on the call. By default, if the terminal is not defined, these will return null. An example of when this would occur is when a phoen goes offhook, and a one-sided call is created. The CalledTerminal would be null in this scenario. The terminal for the called party is only set AFTER the called party answers a call.
Terminal	getCurrentCalledTerminal ()	This method returns a Terminal object that represents the terminal of the called party on the call. By default, if the terminal is not defined, these will return null. An example of when this would occur is when a phoen goes offhook, and a one-sided call is created. The CalledTerminal would be null in this scenario. The terminal for the called party is only set AFTER the called party answers a call.

Interface	Method	Description
java. lang. String	getGlobalizedCallingParty ()	This will return the globalizedCallingParty
CiscoPartyInfo	getLastRedirectedPartyInfo ()	Returns the PartyInfo of the last redirecting party of the call.
boolean	getLastRedirectingAddressPI ()	Returns the Presentation Indicator (PI) that is associated with getLastRedirectingAddressPI.
CiscoPartyInfo	getLastRedirectingPartyInfo ()	Deprecated. - use getLastRedirectedPartyInfo ();
javax. telephony. Address	getModifiedCalledAddress ()	This interface returns the modified called Address for the call if called party is modified by using called party transformation pattern or other means.
javax. telephony. Address	getModifiedCallingAddress ()	This interface returns the modified calling Address for the call if an application modifies its calling party by using the selectRoute API or other means.
boolean	isPersistentCall ()	This interface returns true if the call is a persistent call and false if the call is a normal call.
javax. telephony. Connection[]	startMonitor (javax. telephony. TerminalMonitorInitiatorterminal, javax. telephony. AddressMonitorInitiatoraddress, intmonitorTargetcallid, java. lang. StringmonitorTargetDN, java. lang. StringmonitorTargetTerminalName, intmonitorType, intplayToneDirection)	If the application has the information about the call at the monitor target, the application can use this interface to monitor calls.
javax. telephony. Connection[]	startMonitor (javax. telephony. TerminalMonitorInitiatorterminal, javax. telephony. AddressMonitorInitiatoraddress, javax. telephony. TerminalConnectiontermConnofMonitorTarget, intmonitorType, intPlayToneDirection)	If the application is observing the monitor target (agent) Address, the application can use the Terminal connection of the monitor target (agent) to initiate a monitor request.
javax. telephony. Connection	transfer (java. lang. Stringaddress, java. lang. StringfacCode, java. lang. StringcmcCode)	This method is similar to the CallControlCall. transfer (String address) interface except that it also takes facCode (Forced Authorization Code) and cmcCode (Client Matter Code) if the transfer Address requires these codes to offer the call.
CiscoMultiMedia CapabilityInfo	getCallingTerminalMultiMediaCapabilityInfo ()	Returns the calling party terminal multimedia capability.
CiscoMultiMedia CapabilityInfo	getCalledTerminalMultiMediaCapabilityInfo ()	Returns the called party terminal multimedia capability.

Note In Cisco Unified JTAPI implementation, CallControlCall.getCalledAddress() returns the first called party of the call which is the original called party.

Inherited Methods

From Interface javax.telephony.callcontrol.CallControlCall

addParty, conference, consult, consult, drop, getCalledAddress, getCallingAddress, getCallingTerminal, getConferenceController, getConferenceEnable, getLastRedirectedAddress, getTransferController, getTransferEnable, offHook, setConferenceController, setConferenceEnable, setTransferController, setTransferEnable, transfer, transfer

From Interface javax.telephony.Call

addObserver, connect, getCallCapabilities, getCapabilities, getConnections, getObservers, getProvider, getState, removeObserver

From Interface com.cisco.jtapi.extensions.CiscoObjectContainer

getObject, setObject

Parameters

- origterm -
- origaddr -
- dialedDigits -
- featurePriority -

Conference Controller

For the conferencing feature to happen, a common participant must belong to all the Calls, as represented TerminalConnection of common participants on controller Terminal. These TerminalConnections are known as the conference controllers. At the most, only one of TerminalConnection on the Calls at controller Terminal would be in CallControlTerminalConnection.TALKING state, and hence, the TerminalConnection on the secondary Call should be in the CallControlTerminalConnection.HELD state. As a result of invokation of this method, all the conference controller TerminalConnection merge into one TerminalConnection.

Applications can set which Terminal would acts as the conference controller when a conference call gets set up by setting up Conference controller TerminalConnection via invoking CallControlCalll.setConferenceController() method. The CalControlCall.getConferenceController() method returns the current conference controller, or null if there is none. If no conference controller is set initially, the implementation chooses a suitable TerminalConnection when the conferencing feature is invoked.

Telephone Call Argument

All participants from the secondary Calls, passed as the argument to this method, move to the Call on which this method was invoked. That is, new Connections and TerminalConnections for the participant in the secondary Calls are created on this Call. The Connections and TerminalConnections on the secondary Calls get removed from the Call, and the Call moves to the Call.INVALID state.

Other Shared Participants

There may exist other Addresses and Terminals that are part of some calls in addition to the designated conference controller. In these instances, those participants that are shared between both Calls are merged into one. That is, the Connections and TerminalConnections on this Calls stay unchanged. The corresponding Connections and TerminalConnections on the secondary Calls get removed from that Call.

Pre-Conditions

1. Let tc1 be the conference controller on this Call
2. Let connection1 = tc1.getConnection()
3. Let tc2 to tcN be the conference controllers on otherCalls
4. (this.getProvider()).getState() = = Provider.IN_SERVICE
5. this.getState() = = Call.ACTIVE
6. tc1.getTerminal() = = tc2.getTerminal()... = tcN.getTerminal
7. tc1.getCallControlState() = = CallControlTerminalConnection.TALKING/HELD
8. tc2-tcN.getCallControlState() = = CallControlTerminalConnection.HELD/TALKING
9. this ! = otherCalls

Post-Conditions

1. (this.getProvider()).getState() = = Provider.IN_SERVICE
2. this.getState() = = Call.ACTIVE
3. otherCall.getState() = = INVALID
4. Let c[] be the Connections to be merged from otherCall
5. Let tc[] be the TerminalConnections to be merged from otherCall
6. Let new(c) be the set of new Connections created on this Call
7. Let new(tc) be the set of new TerminalConnections created on this Call
8. new(c) element of this.getConnections()
9. new(c).getCallState() = = c.getCallState()
10. new(tc) element of (this.getConnections()).getTerminalConnections()
11. new(tc).getCallState() = = tc.getCallState()
12. c[i].getCallControlState() = = CallControlConnection.DISCONNECTED for all i
13. tc[i].getCallControlState() = = CallControlTerminalConnection.DROPPED for all i
14. CallInvalidEv is delivered for otherCall
15. CallCtlConnDisconnectedEv/ConnDisconnectedEv is delivered for all c[i]
16. CallCtlTermConnDroppedEv/TermConnDroppedEv is delivered for all tc[i]

17. ConnCreatedEv is delivered for all new(c)
18. TermConnCreatedEv is delivered for all new(tc)
19. Appropriate events are delivered for all new(c) and new(tc)

Parameters

otherCalls - The Other Calls which are to be merged with this Call object.

Throws

javax.telephony.InvalidArgumentException - The Call object that is provided is not valid for the conference.

javax.telephony.InvalidStateException - Thismeans that the Provider is not "in service, " the Call is not "active, " or the conference controllers are not in the proper state.

javax.telephony.MethodNotSupportedException - The implementation does not support this method.

javax.telephony.PrivilegeViolationException - The application does not have the proper authority to invoke this method.

javax.telephony.ResourceUnavailableException - This means that an internal resource that is necessary for the successful invocation of this method is not available.

See Also

ConnCreatedEv, TermConnCreatedEv, ConnDisconnectedEv, TermConnDroppedEv, CallInvalidEv, CallCtlConnDisconnectedEv, CallCtlTermConnDroppedEv

javax.telephony.Connection transfer(java.lang.Stringaddress java.lang.StringfacCode, java.lang.StringcmcCode)

Throws for connect(Terminal, Address, String, CiscoRTPParams)

javax.telephony.InvalidArgumentException, javax.telephony.InvalidStateException, javax.telephony.InvalidPartyException, javax.telephony.MethodNotSupportedException, javax.telephony.PrivilegeViolationException, javax.telephony.ResourceUnavailableExceptionThis method is similar to the CallControlCall.transfer(String address) interface except that it also takes facCode (Forced Authorization Code) and cmcCode (Client Matter Code) if the transfer Address requires these codes to offer the call. If only one of the codes is required, the other code may need to be a null value.

If the user enters no codes, or invalid codes, the call may not be offered and platformException may contain the following error codes:

CiscoJTAPIException.CTIERR_FAC_CMC_REASON_FAC_NEEDED
CiscoJTAPIException.CTIERR_FAC_CMC_REASON_CMC_NEEDED
CiscoJTAPIException.CTIERR_FAC_CMC_REASON_FAC_CMC_NEEDED
CiscoJTAPIException.CTIERR_FAC_CMC_REASON_FAC_INVALID
CiscoJTAPIException.CTIERR_FAC_CMC_REASON_CMC_INVALID

This overloaded version of this method transfers all participants currently on this Call, with the exception of the transfer controller participant, to another Address. This is often called a "single-step transfer" because the transfer feature places another call and performs the transfer simultaneously. The Address string argument to this method must be valid and complete.

The Transfer Controller

The transfer controller for this version of this method represents the participant on this Call around which the transfer is taking place and who drops off the Call after the transfer has completed. The transfer controller is a TerminalConnection that must be in the CallControlTerminalConnection.TALKING state.

Applications may control which TerminalConnection acts as the transfer controller via the CallControlCall.setTransferController() method. The CallControlCall.getTransferController() method returns the current transfer controller, or null if there is none. If no transfer controller is set, the implementation chooses a suitable TerminalConnection when the transfer feature gets invoked.

When the transfer feature gets invoked, the transfer controller moves into the CallControlTerminalConnection.DROPPED state. If it is the only TerminalConnection associated with its Connection, then its Connection moves into the CallControlConnection.DISCONNECTED state as well.

The New Connection

This method creates and returns a new Connection representing the party to which the Call was transferred. This Connection may be null if the Call has been transferred outside of the Provider domain and can no longer be tracked. This Connection must at least be in the CallControlConnection.IDLE state. The Connection state may have progressed beyond "idle" before this method returns, and should be reflected by an event. This new Connection will progress as any normal destination Connection on a call. Typical scenarios for this Connection are described by the Call.connect() method.

Pre-Conditions

1. Let tc be the transfer controller on this Call
2. (this.getProvider()).getState() = = Provider.IN_SERVICE
3. this.getState() = = Call.ACTIVE
4. tc.getCallControlState() = = CallControlTerminalConnection.TALKING

Post-Conditions

1. Let newconnection be the Connection created and returned
2. Let connection = = tc.getConnection()
3. (this.getProvider()).getState() = = Provider.IN_SERVICE
4. this.getState() = = Call.ACTIVE
5. tc.getCallControlState() = = CallControlTerminalConnection.DROPPED
6. If connection.getTerminalConnections().length = = 1, then connection.getCallControlState() = = CallControlConnection.DISCONNECTED
7. newconnection is an element of this.getConnections(), if not null.
8. newconnection.getCallControlState() at least CallControlConnection.IDLE, if not null.
9. ConnCreatedEv is delivered for newconnection
10. CallCtlTermConnDroppedEv/TermConnDroppedEv is delivered for tc

11. CallCtlConnDisconnectedEv/ConnDisconnectedEv is delivered for connection if no other TerminalConnections

Parameters

- address - The destination Address string(dialedDigits) to which the Call is being transferred.
- facCode - The Force Authorization Code
- cmcCode - The Client Matter Code

Returns

The new Connection associated with the destination, or null.

Throws

javax.telephony.InvalidArgumentException - The TerminalConnection provided as controlling the transfer is not valid or not part of this Call.

javax.telephony.InvalidStateException - This means that the Provider is not "in service, " the Call is not "active, " or the transfer controller is not "talking."

javax.telephony.InvalidPartyException - The destination Address is not valid or complete.

javax.telephony.MethodNotSupportedException - The implementation does not support this method.

javax.telephony.PrivilegeViolationException - The application does not have the proper authority to invoke this method.

javax.telephony.ResourceUnavailableException - An internal resource necessary for the successful invocation of this method is unavailable.

See Also

ConnCreatedEv, ConnDisconnectedEv, TermConnDroppedEv, CallCtlConnDisconnectedEv, CallCtlTermConnDroppedEv

getCurrentCalledAddressPIboolean getCurrentCalledAddressPI()Returns the Presentation Indicator(PI) that is associated with CurrentCalledAddress. If it returns true, the application can display this Address name to the end users. If it returns false, the application should not display this Address name to end users.

getCurrentCalledDisplayNamePIboolean getCurrentCalledDisplayNamePI()Returns the Presentation Indicator(PI) that is associated with getCurredCalledDisplayNamePI. If it returns true, the application can display this DisplayName to the end users. If it returns false, the application should not display this DisplayName to the end users.

getCurrentCallingAddressPIboolean getCurrentCallingAddressPI()Returns the Presentation Indicator(PI) that is associated with getCurrentCallingAddressPI. If it returns true, the application can display this Address name to the end users. If it returns false, the application should not display this Address name to the end users.

getCurrentCallingDisplayNamePIboolean getCurrentCallingDisplayNamePI()Returns the Presentation Indicator(PI) that is associated with getCurrentCalledDisplayNamePI. If it returns true, the application can display this DisplayName to the end users. If it returns false, the application should not display this DisplayName to the end users.

getLastRedirectingAddressPIboolean getLastRedirectingAddressPI()Returns the Presentation Indicator(PI) that is associated with getLastRedirectingAddressPI. If it returns true, the application can display this Address name to the end users. If it returns false, the application should not display this Address name to the end users.

getCalledAddressPIboolean getCalledAddressPI()Returns the Presentation Indicator(PI) that is associated with getCalledAddressPI. If it returns true, the application can display this Address name to the end users If it returns false, the application should not display this Address name to the end users.

getCallingAddressPIboolean getCallingAddressPI()Returns the Presentation Indicator(PI) that is associated with getCallingAddressPI. If it returns true, the application can display this Address name to the end users. If it returns false, the application should not display this Address name to the end users.

getCurrentCalledPartyUnicodeDisplayNamejava.lang.String
getCurrentCalledPartyUnicodeDisplayName()Returns the Unicode display name of the called party in the call. It returns null if the display name is unknown.

getCurrentCalledPartyUnicodeDisplayNamelocaleint
getCurrentCalledPartyUnicodeDisplayNamelocale()Returns the locale of the current called party Unicode display name. CiscoLocale interface lists the supported locales.

getCurrentCallingPartyUnicodeDisplayNamejava.lang.String
getCurrentCallingPartyUnicodeDisplayName()Returns the Unicode display name of the calling party in the call. It returns null if the display name is unknown.

getCurrentCallingPartyUnicodeDisplayNamelocaleint
getCurrentCallingPartyUnicodeDisplayNamelocale()Returns the locale of the current called party Unicode display name.

getCurrentCallingPartyInfoCiscoPartyInfo getCurrentCallingPartyInfo()Returns the PartyInfo of the current calling party of the call.

getCurrentCalledPartyInfoCiscoPartyInfo getCurrentCalledPartyInfo()Returns the PartyInfo of the current called party of the call.

getLastRedirectingPartyInfoCiscoPartyInfo getLastRedirectingPartyInfo()Deprecated.- use getLastRedirectedPartyInfo();

Returns the PartyInfo of the last redirecting party of the call.

getLastRedirectedPartyInfoCiscoPartyInfo getLastRedirectedPartyInfo()Returns the PartyInfo of the last redirecting party of the call.

getCalledPartyInfoCiscoPartyInfo getCalledPartyInfo()Returns the PartyInfo of the called party of the call.

javax.telephony.Connection[]startMonitor(javax.telephony.TerminalMonitorInitiatorterminal,
javax.telephony.AddressMonitorInitiatoraddress,
javax.telephony.TerminalConnectiontermConnofMonitorTarget, intmonitorType, intPlayToneDirection)

throws

javax.telephony.ResourceUnavailableException, javax.telephony.PrivilegeViolationException,
javax.telephony.InvalidPartyException, javax.telephony.InvalidArgumentException,
javax.telephony.InvalidStateException, javax.telephony.MethodNotSupportedException

If the application is observing the monitor target (agent) Address, the application can use the Terminal connection of the monitor target (agent) to initiate a monitor request. This interface places a call from an originating endpoint to monitor the call at the monitor target.

Pre-Conditions

1. (this.getProvider()).getState() = = Provider.IN_SERVICE
2. this.getState() = = Call.IDLE
3. ((CiscoProviderCapabilities)(this.getTerminal().getProvider().getProviderCapabilities()).canMonitor() = = TRUE
4. TerminalConnection.getProvider() = = this.getProvider()

Parameters

- MonitorInitiatorterminal - - The originating Terminal
- MonitorInitiatoraddress - - The originating Address
- termConnofMonitorTarget - - The TerminalConnection of the target
- monitorType - - The type of monitor. Use CiscoCall.SILENT_MONITOR.
- PlayToneDirection - - Indicates whether the tone needs to be played to the target, the initiator, or both. This should be one of CiscoCall.PLAYTONE_NOLOCAL_OR_REMOTE, CiscoCall.PLAYTONE_LOCALONLY, CiscoCall.PLAYTONE_REMOTEONLY, or CiscoCall.PLAYTONE_BOTHLOCALANDREMOTE

Throws

javax.telephony.ResourceUnavailableException

javax.telephony.PrivilegeViolationException

javax.telephony.InvalidPartyException

javax.telephony.InvalidArgumentException

javax.telephony.InvalidStateException

javax.telephony.MethodNotSupportedException

Related Documentation

See CallControlCall for more information.

CiscoCallChangedEv

The system delivers the CiscoCallChangedEv event to the call observer for all supported features whenever the Global Call ID (GCID) of the call changes. CiscoCallChangedEv gets delivered when the GCID of the call changes due to path replacement (QSIG_PR) and for other features, including transfer, conference, barge, cbarge, and unpark. In the case of shared lines, multiple CiscoCallChangedEv events get delivered.

The system also delivers this event when two or more calls get merged into one. Transfer, conference, unpark, Barge, and CBarge will trigger this event. Application can invoke CiscoCallEv.getCiscoFeatureReason() to find the feature code that caused this event.

The system reports this event via the CallControlCallObserver interface.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

javax.telephony.events.CallEv, CiscoCallEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoCallChangedEv extends CiscoCallEv

Fields

Table 61: Fields in CiscoCallChangedEv

Interface	Field
static int	ID

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoCallEv

CAUSE_ACCESSINFORMATIONDISCARDED, CAUSE_BARGE, CAUSE_BCBPRESENTLYAVAIL, CAUSE_BCNAUTHORIZED, CAUSE_BEARERCAPNIMPL, CAUSE_CALLBEINGDELIVERED, CAUSE_CALLIDINUSE, CAUSE_CALLMANAGER_FAILURE, CAUSE_CALLREJECTED, CAUSE_CALLSPLIT, CAUSE_CHANTYPENIMPL, CAUSE_CHANUNACCEPTABLE, CAUSE_CTICCMSIP400BADREQUEST, CAUSE_CTICCMSIP401UNAUTHORIZED, CAUSE_CTICCMSIP402PAYMENTREQUIRED, CAUSE_CTICCMSIP403FORBIDDEN, CAUSE_CTICCMSIP404NOTFOUND, CAUSE_CTICCMSIP405METHODNOTALLOWED, CAUSE_CTICCMSIP406NOTACCEPTABLE, CAUSE_CTICCMSIP407PROXYAUTHENTICATIONREQUIRED, CAUSE_CTICCMSIP408REQUESTTIMEOUT, CAUSE_CTICCMSIP410GONE, CAUSE_CTICCMSIP411LENGTHREQUIRED, CAUSE_CTICCMSIP413REQUESTENTITYTOOLONG, CAUSE_CTICCMSIP414REQUESTURITOOLONG, CAUSE_CTICCMSIP415UNSUPPORTEDMEDIATYPE, CAUSE_CTICCMSIP416UNSUPPORTEDURISCHEME, CAUSE_CTICCMSIP420BADEXTENSION, CAUSE_CTICCMSIP421EXTENSTIONREQUIRED, CAUSE_CTICCMSIP423INTERVALTOOBRIEF, CAUSE_CTICCMSIP480TEMPORARILYUNAVAILABLE, CAUSE_CTICCMSIP481CALLLEGDOESNOTEXIST, CAUSE_CTICCMSIP482LOOPDETECTED, CAUSE_CTICCMSIP483TOOMANYHOOPS, CAUSE_CTICCMSIP484ADDRESSINCOMPLETE, CAUSE_CTICCMSIP485AMBIGUOUS, CAUSE_CTICCMSIP486BUSYHERE,

CAUSE_CTICCMSIP487REQUESTTERMINATED, CAUSE_CTICCMSIP488NOTACCEPTABLEHERE, CAUSE_CTICCMSIP491REQUESTPENDING, CAUSE_CTICCMSIP493UNDECIPHERABLE, CAUSE_CTICCMSIP500SERVERINTERNALERROR, CAUSE_CTICCMSIP501NOTIMPLEMENTED, CAUSE_CTICCMSIP502BADGATEWAY, CAUSE_CTICCMSIP503SERVICEUNAVAILABLE, CAUSE_CTICCMSIP504SERVERTIMEOUT, CAUSE_CTICCMSIP505SIPVERSIONNOTSUPPORTED, CAUSE_CTICCMSIP513MESSAGETOOLARGE, CAUSE_CTICCMSIP600BUSYEVERYWHERE, CAUSE_CTICCMSIP603DECLINE, CAUSE_CTICCMSIP604DOESNOTEXISTANYWHERE, CAUSE_CTICCMSIP606NOTACCEPTABLE, CAUSE_CTICONFERENCEFULL, CAUSE_CTIDEVICENOTPREEMPTABLE, CAUSE_CTIDROPCONFEREE, CAUSE_CTIMANAGER_FAILURE, CAUSE_CTIPRECEDENCECALLBLOCKED, CAUSE_CTIPRECEDENCELEVELEXCEEDED, CAUSE_CTIPRECEDENCEOUTOFBANDWIDTH, CAUSE_CTIPREEMPTFORREUSE, CAUSE_CTIPREEMPTNOREUSE, CAUSE_DESTINATIONOUTOFORDER, CAUSE_DESTNUMMISSANDDCNOTSUB, CAUSE_DPARK, CAUSE_DPARK_REMINDER, CAUSE_DPARK_UNPARK, CAUSE_EXCHANGEROUTINGERROR, CAUSE_FAC_CMC, CAUSE_FACILITYREJECTED, CAUSE_IDENTIFIEDCHANDOESNOTEXIST, CAUSE_IENIMPL, CAUSE_INBOUNDBLINDTRANSFER, CAUSE_INBOUNDCONFERENCE, CAUSE_INBOUNDTRANSFER, CAUSE_INCOMINGCALLBARRED, CAUSE_INCOMPATABLEDDESTINATION, CAUSE_INTERWORKINGUNSPECIFIED, CAUSE_INVALIDCALLREFVALUE, CAUSE_INVALIDIECONTENTS, CAUSE_INVALIDMESSAGEUNSPECIFIED, CAUSE_INVALIDNUMBERFORMAT, CAUSE_INVALIDTRANSITNETSEL, CAUSE_MANDATORYIEMISSING, CAUSE_MSGNCOMPATABLEWCS, CAUSE_MSGTYPENCOMPATWCS, CAUSE_MSGTYPENIMPL, CAUSE_NETOUTOFORDER, CAUSE_NOANSWERFROMUSER, CAUSE_NOCALLSUSPENDED, CAUSE_NOCIRCAVAIL, CAUSE_NOERROR, CAUSE_NONSELECTEDUSERCLEARING, CAUSE_NORMALCALLCLEARING, CAUSE_NORMALUNSPECIFIED, CAUSE_NOROUTETODDESTINATION, CAUSE_NOROUTETOTRANSITNET, CAUSE_NOUSERRESPONDING, CAUSE_NUMBERCHANGED, CAUSE_ONLYRDIVEARERCAPAVAIL, CAUSE_OUTBOUNDCONFERENCE, CAUSE_OUTBOUNDTRANSFER, CAUSE_OUTOFBANDWIDTH, CAUSE_PROTOCOLERRORUNSPECIFIED, CAUSE_QSIG_PR, CAUSE_QUALOFSERVNAVAIL, CAUSE_QUIET_CLEAR, CAUSE_RECOVERYONTIMEREXPIRY, CAUSE_REDIRECTED, CAUSE_REQCALLIDHASBEENCLEARED, CAUSE_REQCIRCNAVIL, CAUSE_REQFACILITYNIMPL, CAUSE_REQFACILITYNOTSUBSCRIBED, CAUSE_RESOURCESNAVAIL, CAUSE_RESPONSETOSTATUSENQUIRY, CAUSE_SERVNOTAVAILUNSPECIFIED, CAUSE_SERVOPERATIONVIOLATED, CAUSE_SERVOROPTNAVAILORIMPL, CAUSE_SUBSCRIBERABSENT, CAUSE_SUSPCALLBUTNOTTHISONE, CAUSE_SWITCHINGEQUIPMENTCONGESTION, CAUSE_TEMPORARYFAILURE, CAUSE_UNALLOCATEDNUMBER, CAUSE_USERBUSY

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 62: Methods in CiscoCallChangedEv

Interface	Method	Description
Interface	getConnection()	Returns the CiscoConnection to the Address where the change occurred.
CiscoConnection	getOriginalCall()	Returns the call that will go to INVALID state.
CiscoCall	getSurvivingCall()	Returns the call that will remain active after the callID change.
CiscoCall	getTerminalConnection()	Returns the TerminalConnection where the change occurred. This value could be null if the call ID changes before the TerminalConnection gets created on the Address.

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoCallEv

getCiscoCause, getCiscoFeatureReason

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values for more information.

CiscoCallConsultCancelledEv

This event notifies applications that a cancel operation has been invoked.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	New event for the Swap/Cancel - Transfer/Conference Behavior Change feature.

Superinterfaces

None

Declaration

public interface CiscoCallConsultCancelledEv

Fields

None

Inherited Fields

None

Methods

Table 63: Methods in CiscoCallConsultCancelledEv

Interface	Method	Description
CiscoCall	getConsultCall()	Returns the consult call for which consult operation is cancelled. If the consult call does not exist, it returns NULL. The getCall() API on this call event returns the parent call.

Inherited Methods

None

Related Documentation

None.

CiscoCallCtlConnOfferedEv

The CiscoCallCtlConnOfferedEv interface extends the CallCtlConnOfferedEv interface to let applications obtain the IP Address of the calling party Terminal. The IP Address information might not be available for all calling party devices. A return value of 0 (or null) indicates that the information is not available.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

javax.telephony.callcontrol.events.CallCtlCallEv, javax.telephony.callcontrol.events.CallCtlConnEv, javax.telephony.callcontrol.events.CallCtlConnOfferedEv, javax.telephony.callcontrol.events.CallCtlEv, javax.telephony.events.CallEv, javax.telephony.events.ConnEv, javax.telephony.events.Ev

Declaration

public interface CiscoCallCtlConnOfferedEv extends javax.telephony.callcontrol.events.CallCtlConnOfferedEv

Fields

None

Inherited Fields

From Interface javax.telephony.callcontrol.events.CallCtlConnOfferedEv

None

From Interface javax.telephony.callcontrol.events.CallCtlEv

CAUSE_ALTERNATE, CAUSE_BUSY, CAUSE_CALL_BACK, CAUSE_CALL_NOT_ANSWERED, CAUSE_CALL_PICKUP, CAUSE_CONFERENCE, CAUSE_DO_NOT_DISTURB, CAUSE_PARK, CAUSE_REDIRECTED, CAUSE_REORDER_TONE, CAUSE_TRANSFER, CAUSE_TRUNKS_BUSY, CAUSE_UNHOLD

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 64: Methods in CiscoCallCtlConnOfferedEv

Interface	Method	Description
java.net.InetAddress	getCallingPartyIpAddr()	Returns the IP address of the calling party, or 0 (or null) if the IP Address is not available.

Inherited Methods

From Interface javax.telephony.callcontrol.events.CallCtlCallEv

getCalledAddress, getCallingAddress, getCallingTerminal, getLastRedirectedAddress

From Interface javax.telephony.callcontrol.events.CallCtlEv

getCallControlCause

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ConnEv

getConnection

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

None

CiscoCallCtlTermConnHeldReversionEv

The CiscoCallCtlTermConnHeldReversionEv event indicates that hold reversion notification has been received on the TerminalConnection from Cisco Unified Communications Manager.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

javax.telephony.callcontrol.events.CallCtlCallEv, javax.telephony.callcontrol.events.CallCtlEv, javax.telephony.callcontrol.events.CallCtlTermConnEv, javax.telephony.events.CallEv, javax.telephony.events.Ev, javax.telephony.events.TermConnEv

Declaration

public interface CiscoCallCtlTermConnHeldReversionEv extends javax.telephony.callcontrol.events.CallCtlTermConnEv

Fields

Table 65: Fields in CiscoCallCtlTermConnHeldReversionEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.callcontrol.events.CallCtlEv

CAUSE_ALTERNATE, CAUSE_BUSY, CAUSE_CALL_BACK, CAUSE_CALL_NOT_ANSWERED, CAUSE_CALL_PICKUP, CAUSE_CONFERENCE, CAUSE_DO_NOT_DISTURB, CAUSE_PARK, CAUSE_REDIRECTED, CAUSE_REORDER_TONE, CAUSE_TRANSFER, CAUSE_TRUNKS_BUSY, CAUSE_UNHOLD

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.callcontrol.events.CallCtlCallEv

getCalledAddress, getCallingAddress, getCallingTerminal, getLastRedirectedAddress

From Interface javax.telephony.callcontrol.events.CallCtlEv

getCallControlCause

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermConnEv

getTerminalConnection

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values for more information.

CiscoCallEv

The CiscoCallEv interface, which extends the JTAPI core javax.telephony.events.CallEv interface, serves as the base interface for all Cisco-extended JTAPI Call events. Every Call-related event in this package extends this interface, directly or indirectly.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

javax.telephony.events.CallEv, CiscoEv, javax.telephony.events.Ev

Subinterfaces

CiscoCallChangedEv, CiscoCallSecurityStatusChangedEv, CiscoConferenceChainAddedEv, CiscoConferenceChainRemovedEv, CiscoConferenceEndEv, CiscoConferenceStartEv, CiscoConsultCallActiveEv, CiscoToneChangedEv, CiscoTransferEndEv, CiscoTransferStartEv

Declaration

public interface CiscoCallEv extends CiscoEv, javax.telephony.events.CallEv

Fields

Table 66: Fields in CiscoCallEv

Interface	Field	Description
Static int	CAUSE_ACCESSINFORMATIONDISCARDED	This cause indicates that the network could not deliver access information to the remote user as requested.
Static int	CAUSE_BARGE	It indicates the call is a BARGE call.
staticint	CAUSE_BCBPRESENTLYAVAIL	This cause indicates that the user has requested a bearer capability which is implemented by the equipment which generated this cause but which is not available at this time.
static int	CAUSE_BCNAUTHORIZED	This cause indicates that the user has requested a bearer capability which is implemented by the equipment which generated this cause but the user is not authorized to use.
static int	CAUSE_BEARERCAPNIMPL	This cause indicates that the equipment sending this cause does not support the bearer capability requested.
static int	CAUSE_CALLBEINGDELIVERED	This cause indicates that the user has been awarded the incoming call and that the incoming call is being connected to a channel already established to that user for similar calls.
static int	CAUSE_CALLIDINUSE	This cause indicates that the network has received a call suspended request containing a call identity (including the null call identity) which is already in use for a suspended call within the domain of interfaces over which the call might be resumed.
static int	CAUSE_CALLMANAGER_FAILURE	This cause indicates the failure due to CALL Manager Failure.

Interface	Field	Description
static int	CAUSE_CALLREJECTED	This cause indicates that the equipment sending this cause does not wish to accept this call.
static int	CAUSE_CALLSPLIT	This cause indicates the call split, it could mean conference or transfer.
static int	CAUSE_CHANTYPENIMPL	This cause indicates that the equipment sending this cause does not support the channel type requested.
static int	CAUSE_CHANUNACCEPTABLE	This cause indicates that the channel most recently identified is not acceptable to the sending entity for use in this call.
static int	CAUSE_CTICCMSIP400BADREQUEST	This cause indicates the call is rejected due to bad request.
static int	CAUSE_CTICCMSIP401UNAUTHORIZED	This cause indicates the request is valid but is not authorized.
static int	CAUSE_CTICCMSIP402PAYMENTREQUIRED	This cause indicates the payment is required for usage.
static int	CAUSE_CTICCMSIP403FORBIDDEN	This cause indicates the server understood the request, but is refusing to fulfill it..
static int	CAUSE_CTICCMSIP404NOTFOUND	This cause indicates the request URI cannot be located by the server.
static int	CAUSE_CTICCMSIP405METHODNOTALLOWED	This cause indicates the method specified in the Request-Line is understood, but not allowed for the address identified by the Request-URI.
static int	CAUSE_CTICCMSIP406NOTACCEPTABLE	This cause indicates the request cannot be proccessed due to requirements in the request cannot be met.
static int	CAUSE_CTICCMSIP407PROXY AUTHENTICATIONREQUIRED	This cause indicates that requset is not authorized and proxy authentication is required for the operation.
static int	CAUSE_CTICCMSIP408REQUESTTIMEOUT	This cause indicates the time out error for the request.
static int	CAUSE_CTICCMSIP410GONE	This cause indicates the requested resource is no longer available at the server and no forwarding address is known.
static int	CAUSE_CTICCMSIP411LENGTHREQUIRED	This cause indicates that an interworking message length is required.
static int	CAUSE_CTICCMSIP413REQUESTENTITY TOOLONG	This cause indicates that the server is refusing to process a request because the request entity-body is larger than the server is willing or able to process.
static int	CAUSE_CTICCMSIP414REQUESTURI TOOLONG	This cause indicates that the server is refusing to service the request because the Request-URI is longer than the server is willing to interpret.

Interface	Field	Description
static int	CAUSE_CTICCMSIP415UNSUPPORTED MEDIATYPE	This cause indicates the server is refusing to service the request because the message body of the request indicates the Media Type which is not supported by the server for the requested method.
static int	CAUSE_CTICCMSIP416UNSUPPORTED URISCHEME	This cause indicates the server cannot process the request because the scheme of the URI in the Request-URI is unknown to the server.
static int	CAUSE_CTICCMSIP420BADEXTENSION	This cause indicates the server did not understand the protocol extension specified in a Proxy-Require or Require header field.
static int	CAUSE_CTICCMSIP421EXTENSTION REQUIRED	This cause indicates the UAS needs a particular extension to process the request, but this extension is not listed in a Supported header field in the request.
static int	CAUSE_CTICCMSIP423INTERVALTOOBRIEF	This cause indicates that the server is rejecting the request because the expiration time of the resource refreshed by the request is too short.
static int	CAUSE_CTICCMSIP480TEMPORARILY UNAVAILABLE	This cause indicates the callee's end system was contacted successfully but the callee is currently unavailable (for example, is not logged in, logged in but in a state that precludes communication with the callee, or has activated the "do not disturb" feature).
static int	CAUSE_CTICCMSIP481CALLLEGDOES NOTEXIST	This cause indicates the the UAS received a request that does not match any existing dialog or transaction.
static int	CAUSE_CTICCMSIP482LOOPDETECTED	This cause indicates that the server has detected a loop.
static int	CAUSE_CTICCMSIP483TOOMANYHOOPS	This cause indicates the server received a request that contains a Max-Forwards header field with the value zero (or less than actual hops).
static int	CAUSE_CTICCMSIP484ADDRESS INCOMPLETE	This cause indicates that the server received a request with a Request-URI that was incomplete.
static int	CAUSE_CTICCMSIP485AMBIGUOUS	This cause indicates that the Request-URI was ambiguous.
static int	CAUSE_CTICCMSIP486BUSYHERE	This indicates that the callee's end system was contacted successfully, but the callee is currently not willing or able to take additional calls at this end system.
static int	CAUSE_CTICCMSIP487REQUEST TERMINATED	This cause indicates the request was terminated by a BYE or CANCEL request.

Interface	Field	Description
static int	CAUSE_CTICCMSIP488NOTACCEPTABLE HERE	This cause indicates the same meaning as 606 (Not Acceptable), but only applies to the specific resource addressed by the Request-URI and the request may succeed elsewhere.
static int	CAUSE_CTICCMSIP491REQUESTPENDING	This cause indicates the request was received by a UAS that had a pending request within the same dialog.
static int	CAUSE_CTICCMSIP493UNDECIPHERABLE	This cause indicates that the request was received by a UAS that contained an encrypted MIME body for which the recipient does not possess or will not provide an appropriate decryption key.
static int	CAUSE_CTICCMSIP500SERVERINTERNAL ERROR	This cause indicates the server encountered an unexpected condition that prevented it from fulfilling the request.
static int	CAUSE_CTICCMSIP501NOTIMPLEMENTED	This cause indicates the server does not support the functionality required to fulfill the request.
static int	CAUSE_CTICCMSIP502BADGATEWAY	This cause indicates the server, while acting as a gateway or proxy, received an invalid response from the downstream server it accessed in attempting to fulfill the request.
static int	CAUSE_CTICCMSIP503SERVICEUNAVAILABLE	This cause indicates the server is temporarily unable to process the request due to a temporary overloading or maintenance of the server.
static int	CAUSE_CTICCMSIP504SERVERTIMEOUT	This cause indicates the server did not receive a timely response from an external server it accessed in attempting to process the request.
static int	CAUSE_CTICCMSIP505SIPVERSIONNOT SUPPORTED	This cause indicates the server does not support, or refuses to support, the SIP protocol version that was used in the request.
static int	CAUSE_CTICCMSIP513MESSAGETOOLARGE	This cause indicates the server was unable to process the request since the message length exceeded its capabilities.
static int	CAUSE_CTICCMSIP600BUSYEVERYWHERE	This cause indicates the callee's end system was contacted successfully but the callee is busy and does not wish to take the call at this time.
static int	CAUSE_CTICCMSIP603DECLINE	This cause indicates the callee's machine was successfully contacted but the user explicitly does not wish to or cannot participate.
static int	CAUSE_CTICCMSIP604DOESNOTEXIST ANYWHERE	This cause indicates the server has authoritative information that the user indicated in the Request-URI does not exist anywhere.

Interface	Field	Description
static int	CAUSE_CTICCMSIP606NOTACCEPTABLE	This cause indicates the user's agent was contacted successfully but some aspects of the session description such as the requested media, bandwidth, or addressing style were not acceptable.
static int	CAUSE_CTICONFERENCEFULL	This cause indicates the Conference Call is full and no more participants can be added to it.
static int	CAUSE_CTIDEVICENOTPREEMPTABLE	This cause indicates that the device cannot be preempted.
static int	CAUSE_CTIDROPCONFEREE	This cause indicates the disconnection because the party was dropped from conference.
static int	CAUSE_CTIMANAGER_FAILURE	This cause indicates the failure due to CTI Manager Failure.
static int	CAUSE_CTIPRECEDENCECALLBLOCKED	This cause indicates that there are no predictable circuits or that the called user is busy with a call of equal or higher preventable level.
static int	CAUSE_CTIPRECEDENCELEVELEXCEEDED	This cause indicates that the precedence level of the call has exceeded the authorized level.
static int	CAUSE_CTIPRECEDENCEOUTOFBANDWIDTH	This cause indicates the precedence call has hit low bandwidth and cannot proceed.
static int	CAUSE_CTIPREEMPTFORREUSE	This cause indicates that the call is being preempted and the circuit is reserved for reuse by the preempting exchange.
static int	CAUSE_CTIPREEMPTNOREUSE	This cause indicates the call is being preempted.
static int	CAUSE_DESTINATIONOUTOFORDER	This cause indicates that the destination indicated by the user cannot be reached because the interface to the destination is not functioning correctly.
static int	CAUSE_DESTNUMMISSANDDCNOTSUB	This cause indicates that the specified CUG does not exist.
static int	CAUSE_DPARK	It indicates the call is Directed-Parked call.
static int	CAUSE_DPARK_REMINDER	It indicates the call is Directed Park Reminder call.
static int	CAUSE_DPARK_UNPARK	It indicates that Directed Parked call is now unparked.
static int	CAUSE_EXCHANGEROUTINGERROR	This cause indicates that the exchange couldnt route the call to specified destination.
static int	CAUSE_FAC_CMC	It indicates the FAC(Force Authorization Code) or CMC(Client Matter Code) is needed to route the call.
static int	CAUSE_FACILITYREJECTED	This cause is returned when a supplementary service requested by the user cannot be provided by the network.

Interface	Field	Description
static int	CAUSE_IDENTIFIEDCHANDOESNOTEXIST	This cause indicates that the equipment sending this cause has received a request to use a channel not activated on the interface for a call.
static int	CAUSE_IENIMPL	This cause indicates that the equipment sending this cause has received a message which includes information element(s)/parameter(s) not recognized because the information element(s)/parameter name(s) are not defined or are defined but not implemented by the equipment sending the cause.
static int	CAUSE_INBOUNDBLINDTRANSFER	It indicates the call is IN bound Blind Transfer call.
static int	CAUSE_INBOUNDCONFERENCE	It indicates the call is IN bound Conference call.
static int	CAUSE_INBOUNDTRANSFER	It indicates the call is IN bound Transfer call.
static int	CAUSE_INCOMINGCALLBARRED	This cause indicates that the incoming calls for that number is barred.
static int	CAUSE_INCOMPATABLEDDESTINATION	This cause indicates that the equipment sending this cause has received a request to establish a call which has low layer compatibility.
static int	CAUSE_INTERWORKINGUNSPECIFIED	This cause indicates that an interworking call has ended.
static int	CAUSE_INVALIDCALLREFVALUE	This cause indicates that the equipment sending this cause has received a message with a call reference which is not currently in use on the user-network interface.
static int	CAUSE_INVALIDIECONTENTS	This cause indicates that the equipment sending this cause has received and information element which it has implemented; however, one or more of the fields in the information element are coded in such a way which has not been implemented by the equipment sending this cause.
static int	CAUSE_INVALIDMESSAGEUNSPECIFIED	This cause is used to report an invalid message event only when no other cause in the invalid message class applies.
static int	CAUSE_INVALIDNUMBERFORMAT	This cause indicates that the called party cannot be reached because the called party number is not in a valid format or is not complete.
static int	CAUSE_INVALIDTRANSITNETSEL	This cause indicates that a transit network identification was received which is of an incorrect format.
static int	CAUSE_MANDATORYIEMISSING	This cause indicates that the equipment sending this cause has received a message which is missing an information element which must be present in the message before that message can be processed.

Interface	Field	Description
static int	CAUSE_MSGNCOMPATABLEWCS	This cause indicates that a message has been received which is incompatible with the call state.
static int	CAUSE_MSGTYPENCOMPATWCS	This cause indicates that the equipment sending this cause has received a message such that the procedures do not indicate that this is a permissible message to receive while in the call state, or a STATUS message was received indicating an incompatible call state.
static int	CAUSE_MSGTYPENIMPL	This cause indicates that the equipment sending this cause has received a message with a message type it does not recognize either because this is a message not defined or defined but not implemented by the equipment sending this cause.
static int	CAUSE_NETOUTOFORDER	This cause indicates that the network is not functioning correctly and that the condition is likely to last a relatively long period of time.
static int	CAUSE_NOANSWERFROMUSER	This cause is used when the called party has been alerted but does not respond with a connect indication within a prescribed period of time.
static int	CAUSE_NOCALLSUSPENDED	This cause indicates that the network has received a call resume request containing a call identity information element which presently does not indicate any suspended call within the domain of interfaces over which calls may be resumed.
static int	CAUSE_NOCIRCAVAIL	This cause indicates that there is no appropriate circuit/channel presently available to handle the call.
static int	CAUSE_NOERROR	This is usually given when there is no error and operation completes successfuly.
static int	CAUSE_NONSELECTEDUSERCLEARING	This cause indicates that the user has not been awarded the incoming call.
static int	CAUSE_NORMALCALLCLEARING	This cause indicates that the call is being cleared because one of the users involved in the call has requested that the call be cleared.
static int	CAUSE_NORMALUNSPECIFIED	This cause is used to report a normal event only when no other cause in the normal class applies.
static int	CAUSE_NOROUTETODDESTINATION	This cause indicates that the called party cannot be reached because the network through which the call has been routed does not serve the destination desired.
static int	CAUSE_NOROUTETOTRANSITNET	This cause indicates that the equipment sending this cause has received a request to route the call through a particular transit network which it does not recognize.

Interface	Field	Description
static int	CAUSE_NOUSERRESPONDING	This cause is used when a called party does not respond to a call establishment message with either an alerting or connect indication within the prescribed period of time allocated.
static int	CAUSE_NUMBERCHANGED	This cause is returned to a calling party when the called party number indicated by the calling party is no longer assigned.
static int	CAUSE_ONLYRDIVEARERCAPAVAIL	This cause indicates that the calling party has requested an unrestricted bearer service but the equipment sending this cause only supports the restricted version of the requested bearer capability.
static int	CAUSE_OUTBOUNDCONFERENCE	It indicates the call is OUT bound Conference call.
static int	CAUSE_OUTBOUNDTRANSFER	It indicates the call is OUT bound Transfer call
static int	CAUSE_OUTOFBANDWIDTH	This cause indicates that the call could not proceed because of Low Bandwidth.
static int	CAUSE_PROTOCOLERRORUNSPECIFIED	This cause is used to report a protocol error event only when no other cause in the protocol error class applies.
static int	CAUSE_QSIG_PR	It indicates the QSIG Path Replacement in the call.
static int	CAUSE_QUALOFSERVNAVAIL	This cause is used to report that the requested Quality of Service, as defined in Recommendation X.213.
static int	CAUSE_QUIET_CLEAR	It indicates the Call is cleared as Call Manager has gone down, but media between endpoints remain connected.
static int	CAUSE_RECOVERYONTIMEREXPIRY	This cause indicates that a procedure has been initiated by the expiration of a timer in association with error handling procedures.
static int	CAUSE_REDIRECTED	This cause indicates the call is being redirected to different party.
static int	CAUSE_REQCALLIDHASBEENCLEARED	This cause indicates that the network has received a call resume request containing a call identity information element indicating a suspended call that has in the meantime been cleared while suspended (either by network time-out or by the remote user).
static int	CAUSE_REQCIRCNAVIL	This cause is returned when the circuit or channel indicated by the requesting entity cannot be provided by the other side of the interface.
static int	CAUSE_REQFACILITYNIMPL	This cause indicates that the equipment sending this cause does not support the requested.

Interface	Field	Description
static int	CAUSE_REQFACILITYNOTSUBSCRIBED	This cause indicates that the user has requested a supplementary service which is implemented by the equipment which generated this cause but the user is not authorized to use.
static int	CAUSE_RESOURCESNAVAIL	This cause is used to report a resource unavailable event.
static int	CAUSE_RESPONSETOSTATUSENQUIRY	This cause is included in the STATUS message when the reason for generating the STATUS message was the prior receipt of a STATUS INQUIRY.
static int	CAUSE_SERVNOTAVAILUNSPECIFIED	This cause is used to report a service or option not available event only when no other cause in the service or option not available class applies.
static int	CAUSE_SERVOPERATIONVIOLATED	This cause indicates that although the calling party is a member of the CUG for the outgoing CUG call.
static int	CAUSE_SERVOROPTNAVAILORIMPL	This cause is used to report a service or option not implemented event only when no other cause in the service or option not implemented class applies.
static int	CAUSE_SUBSCRIBERABSENT	This cause value is used when a mobile station has logged off.
static int	CAUSE_SUSPCALLBUTNOTTHISONE	This cause indicates that a call resume has been attempted with a call identity which differs from that in use for any presently suspended call(s).
static int	CAUSE_SWITCHINGEQUIPMENTCONGESTION	This cause indicates that the switching equipment generating this cause is experiencing a period of high traffic.
static int	CAUSE_TEMPORARYFAILURE	This cause indicates that the network is not functioning correctly and that the condition is not likely to last a long period of time; e.g., the user may wish to try another call attempt almost immediately.
static int	CAUSE_UNALLOCATEDNUMBER	This cause indicates that the destination requested by the calling user cannot be reached because, it is an invalid number.
static int	CAUSE_USERBUSY	This cause is used to indicate that the called party is unable to accept another call because the user busy condition has been encountered.

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Methods

Table 67: Methods in CiscoCallEv

Interface	Method	Description
Int	getCiscoCause()	Returns the Cisco Unified Communications Manager cause for this event. To function properly, some applications need to know the reason why an event happened at an endpoint that the application is observing. For example, a Connection may be disconnected because the call was not answered (CAUSE_NOANSWERFROMUSER), or whether the caller it was disconnected because it was rejected (CAUSE_CALLREJECTED). Returns: The Cisco Unified Communications Manager cause for this event

Interface	Method	Description
Int	getCiscoFeatureReason()	Returns the Cisco Unified Communications Manager Feature Reason for this event. To function properly, some applications need to know the reason why an event happened. This interface provides the CiscoFeatureReason in JTAPI Call events for current and new features. Existing features, such as transfer, continue to receive the CiscoCause provided by the older interface CiscoCallEv.getCiscoCause(), while this interface will provide REASON_TRANSFER for transfer. Caution: Applications should make sure to handle unrecognized reasons and provide default behavior, because new reasons could be added in the future and this interface may not be backward compatible. The possible values are defined in the CiscoFeatureReason interface. Returns: The Cisco Unified Communications Manager Feature Reason for this event

Related Documentation

See Constant Field Values and CallEv for more information.

CiscoCallFeatureCancelledEv

This event notifies applications that the cancel operation has been invoked

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(2)	Created history table to track changes.

Declaration

public interface CiscoCallFeatureCancelledEv

Methods

Table 68: Methods in CiscoCallFeatureCancelledEv

Interface	Method	Description
CiscoCall	getConsultCall()	Returns the Consult Call for which consult operation is cancelled, if the consult call doesn't exist it will return NULL.

Related Documentation

See Constant Field Values.

CiscoCallID

The CiscoCallID object represents a unique object that is associated with each call. Applications may use the object itself or the integer representation of the object that the intValue() method returns.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

CiscoObjectContainer

Declaration

Public interface CiscoCallID extends CiscoObjectContainer

Fields

None

Methods

Table 69: Methods in CiscoCallID

Interface	Method	Description
Int	intValue()	Returns an integer representation of this object. Returns: Int An integer representation of this object
CiscoCall	getCall()	Returns the CiscoCall corresponding to this CiscoCallID.
int	getCallManagerID()	Returns the Cisco Unified Communications Manager NodeID of the call associated with this CiscoCallID.
int	getGlobalCallID()	Returns the GlobalCallID of the call associated with this CiscoCallID.

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoObjectContainer

getObject, setObject

Related Documentation

None

CiscoMediaCallSecurityIndicator

CiscoMediaCallSecurityIndicator lets you retrieve the security indicator for a call.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoMediaCallSecurityIndicator

Fields

None

Methods

Table 70: Methods in CiscoMediaCallSecurityIndicator

Interface	Method	Description
CiscoCallID	getCallID()	Returns the CiscoCallID.
int	getCiscoMediaSecurityIndicator()	Returns the media security indicator, one of the following constants: CiscoMediaSecurityIndicator. MEDIA_ENCRYPT_USER_NOT_AUTHORIZED CiscoMediaSecurityIndicator. MEDIA_ENCRYPTED_KEYS_UNAVAILABLE CiscoMediaSecurityIndicator. MEDIA_NOT_ENCRYPTED

Interface	Method	Description
CiscoRTPHandle	getCiscoRTPHandle()	Returns a CiscoRTPHandle object.Applications can get a call reference by using CiscoProvider.getCall. If there is no call observer or there was no call observer when this event was delivered, CiscoProvider.getCall may return null.

Related Documentation

See CiscoRTPParams.

CiscoCallSecurityStatusChangedEv

Applications receive CiscoCallSecurityStatusChangedEv when the overall Call security status changes.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

javax.telephony.events.CallEv, CiscoCallEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoCallSecurityStatusChangedEv extends CiscoCallEv

Fields

Table 71: Fields in CiscoCallSecurityStatusChangedEv

Interface	Field
Static int	ID

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoCallEv

CAUSE_ACCESSINFORMATIONDISCARDED, CAUSE_BARGE, CAUSE_BCBPRESENTLYAVAIL, CAUSE_BCNAUTHORIZED, CAUSE_BEARERCAPNIMPL, CAUSE_CALLBEINGDELIVERED, CAUSE_CALLIDINUSE, CAUSE_CALLMANAGER_FAILURE, CAUSE_CALLREJECTED,

CAUSE_CALLSPLIT, CAUSE_CHANTYPENIMPL, CAUSE_CHANUNACCEPTABLE,
CAUSE_CTICCMSIP400BADREQUEST, CAUSE_CTICCMSIP401UNAUTHORIZED,
CAUSE_CTICCMSIP402PAYMENTREQUIRED, CAUSE_CTICCMSIP403FORBIDDEN,
CAUSE_CTICCMSIP404NOTFOUND, CAUSE_CTICCMSIP405METHODNOTALLOWED,
CAUSE_CTICCMSIP406NOTACCEPTABLE,
CAUSE_CTICCMSIP407PROXYAUTHENTICATIONREQUIRED,
CAUSE_CTICCMSIP408REQUESTTIMEOUT, CAUSE_CTICCMSIP410GONE,
CAUSE_CTICCMSIP411LENGTHREQUIRED, CAUSE_CTICCMSIP413REQUESTENTITYTOOLONG,
CAUSE_CTICCMSIP414REQUESTURITOOLONG,
CAUSE_CTICCMSIP415UNSUPPORTEDMEDIATYPE,
CAUSE_CTICCMSIP416UNSUPPORTEDURISCHEME, CAUSE_CTICCMSIP420BADEXTENSION,
CAUSE_CTICCMSIP421EXTENSTIONREQUIRED, CAUSE_CTICCMSIP423INTERVALTOOBRIEF,
CAUSE_CTICCMSIP480TEMPORARILYUNAVAILABLE,
CAUSE_CTICCMSIP481CALLLEGDOESNOTEXIST, CAUSE_CTICCMSIP482LOOPDETECTED,
CAUSE_CTICCMSIP483TOOMANYHOOPS, CAUSE_CTICCMSIP484ADDRESSINCOMPLETE,
CAUSE_CTICCMSIP485AMBIGUOUS, CAUSE_CTICCMSIP486BUSYHERE,
CAUSE_CTICCMSIP487REQUESTTERMINATED, CAUSE_CTICCMSIP488NOTACCEPTABLEHERE,
CAUSE_CTICCMSIP491REQUESTPENDING, CAUSE_CTICCMSIP493UNDECIPHERABLE,
CAUSE_CTICCMSIP500SERVERINTERNALERROR, CAUSE_CTICCMSIP501NOTIMPLEMENTED,
CAUSE_CTICCMSIP502BADGATEWAY, CAUSE_CTICCMSIP503SERVICEUNAVAILABLE,
CAUSE_CTICCMSIP504SERVERTIMEOUT, CAUSE_CTICCMSIP505SIPVERSIONNOTSUPPORTED,
CAUSE_CTICCMSIP513MESSAGETOOLARGE, CAUSE_CTICCMSIP600BUSYEVERYWHERE,
CAUSE_CTICCMSIP603DECLINE, CAUSE_CTICCMSIP604DOESNOTEXISTANYWHERE,
CAUSE_CTICCMSIP606NOTACCEPTABLE, CAUSE_CTICONFERENCEFULL,
CAUSE_CTIDEVICENOTPREEMPTABLE, CAUSE_CTIDROPCONFEREE,
CAUSE_CTIMANAGER_FAILURE, CAUSE_CTIPRECEDENCECALLBLOCKED,
CAUSE_CTIPRECEDENCELEVELEXCEEDED, CAUSE_CTIPRECEDENCEOUTOFBANDWIDTH,
CAUSE_CTIPREEMPTFORREUSE, CAUSE_CTIPREEMPTNOREUSE,
CAUSE_DESTINATIONOUTOFORDER, CAUSE_DESTNUMMISSANDDCNOTSUB, CAUSE_DPARK,
CAUSE_DPARK_REMINDER, CAUSE_DPARK_UNPARK, CAUSE_EXCHANGEROUTINGERROR,
CAUSE_FAC_CMC, CAUSE_FACILITYREJECTED, CAUSE_IDENTIFIEDCHANDOESNOTEXIST,
CAUSE_IENIMPL, CAUSE_INBOUNDBLINDTRANSFER, CAUSE_INBOUNDCONFERENCE,
CAUSE_INBOUNDTRANSFER, CAUSE_INCOMINGCALLBARRED,
CAUSE_INCOMPATABLEDDESTINATION, CAUSE_INTERWORKINGUNSPECIFIED,
CAUSE_INVALIDCALLREFVALUE, CAUSE_INVALIDIECONTENTS,
CAUSE_INVALIDMESSAGEUNSPECIFIED, CAUSE_INVALIDNUMBERFORMAT,
CAUSE_INVALIDTRANSITNETSEL, CAUSE_MANDATORYIEMISSING,
CAUSE_MSGNCOMPATABLEWCS, CAUSE_MSGTYPENCOMPATWCS, CAUSE_MSGTYPENIMPL,
CAUSE_NETOUTOFORDER, CAUSE_NOANSWERFROMUSER, CAUSE_NOCALLSUSPENDED,
CAUSE_NOCIRCAVAIL, CAUSE_NOERROR, CAUSE_NONSELECTEDUSERCLEARING,
CAUSE_NORMALCALLCLEARING, CAUSE_NORMALUNSPECIFIED,
CAUSE_NOROUTETODDESTINATION, CAUSE_NOROUTETOTRANSITNET,
CAUSE_NOUSERRESPONDING, CAUSE_NUMBERCHANGED,
CAUSE_ONLYRDIVEARERCAPAVAIL, CAUSE_OUTBOUNDCONFERENCE,
CAUSE_OUTBOUNDTRANSFER, CAUSE_OUTOFBANDWIDTH,
CAUSE_PROTOCOLERRORUNSPECIFIED, CAUSE_QSIG_PR, CAUSE_QUALOFSERVNAVAIL,
CAUSE_QUIET_CLEAR, CAUSE_RECOVERYONTIMEREXPIRY, CAUSE_REDIRECTED,
CAUSE_REQCALLIDHASBEENCLEARED, CAUSE_REQCIRCNAVIL, CAUSE_REQFACILITYNIMPL,
CAUSE_REQFACILITYNOTSUBSCRIBED, CAUSE_RESOURCESNAVAIL,
CAUSE_RESPONSETOSTATUSENQUIRY, CAUSE_SERVNOTAVAILUNSPECIFIED,
CAUSE_SERVOPERATIONVIOLATED, CAUSE_SERVOROPTNAVAILORIMPL,

CAUSE_SUBSCRIBERABSENT, CAUSE_SUSPCALLBUTNOTTHISONE, CAUSE_SWITCHINGEQUIPMENTCONGESTION, CAUSE_TEMPORARYFAILURE, CAUSE_UNALLOCATEDNUMBER, CAUSE_USERBUSY

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 72: Methods in CiscoCallSecurityStatusChangedEv

Interface	Method	Description
javax.telephony.events.Ev	getID()	Specified by: getID in interface javax.telephony.events.Ev
getCallSecurityStatus	getCallSecurityStatus()	Returns the call security status. This interface can return: CiscoCall.CALLSECURITY_UNKNOWN, CiscoCall.CALLSECURITY_NOTAUTHENTICATED, CiscoCall.CALLSECURITY_AUTHENTICATED, CiscoCall.CALLSECURITY_ENCRYPTED

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoCallEv

getCiscoCause, getCiscoFeatureReason

From Interface javax.telephony.events.Ev

getCause, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See also Constant Field Values for more information.

CiscoConferenceChain

This interface provides links to conference chain connections for the conference calls that are linked together in a conference chain. You can obtain this object from CiscoConferenceChainAddedEv and CiscoConferenceChainRemovedEv.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Declaration

public interface CiscoConferenceChain

Fields

None

Methods

Table 73: Methods in CiscoConferenceChain

Interface	Method	Description
javax.telephony.Connection[]	getChainedConferenceConnections()	Returns an array of Connections for Conference Calls that are chained together in a single conference. Applications can use this list to get all the Conference Calls that are linked together. To get the list of Connections for all the Calls that are chained together in the Conference, the provider must have an observer on at least one party in every conference call.

Interface	Method	Description
CiscoCall[]	getChainedConferenceCalls()	Returns an array of Calls that are chained together in a single conference. This interface returns only the Calls in the conference chain that are observed in the provider.

Related Documentation

See CiscoConferenceChainAddedEv and CiscoConferenceChainRemovedEv for more information.

CiscoConferenceChainAddedEv

The system sends a CiscoConferenceChainAddedEv event when a conference chain connection gets added to a call. This event gets sent every time a new conference chain connection gets added. This event gets reported via theCallControlCallObserver interface.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

All Superinterfaces

javax.telephony.events.CallEv, CiscoCallEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoConferenceChainAddedEv extends CiscoCallEv

Fields

Table 74: Fields in CiscoConferenceChainAddedEv

Interface	Field
static int	ID

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoCallEv

CAUSE_ACCESSINFORMATIONDISCARDED, CAUSE_BARGE, CAUSE_BCBPRESENTLYAVAIL, CAUSE_BCNAUTHORIZED, CAUSE_BEARERCAPNIMPL, CAUSE_CALLBEINGDELIVERED, CAUSE_CALLIDINUSE, CAUSE_CALLMANAGER_FAILURE, CAUSE_CALLREJECTED, CAUSE_CALLSPLIT, CAUSE_CHANTYPENIMPL, CAUSE_CHANUNACCEPTABLE, CAUSE_CTICCMSIP400BADREQUEST, CAUSE_CTICCMSIP401UNAUTHORIZED, CAUSE_CTICCMSIP402PAYMENTREQUIRED, CAUSE_CTICCMSIP403FORBIDDEN, CAUSE_CTICCMSIP404NOTFOUND, CAUSE_CTICCMSIP405METHODNOTALLOWED, CAUSE_CTICCMSIP406NOTACCEPTABLE, CAUSE_CTICCMSIP407PROXYAUTHENTICATIONREQUIRED, CAUSE_CTICCMSIP408REQUESTTIMEOUT, CAUSE_CTICCMSIP410GONE, CAUSE_CTICCMSIP411LENGTHREQUIRED, CAUSE_CTICCMSIP413REQUESTENTITYTOOLONG, CAUSE_CTICCMSIP414REQUESTURITOOLONG, CAUSE_CTICCMSIP415UNSUPPORTEDMEDIATYPE, CAUSE_CTICCMSIP416UNSUPPORTEDURISCHEME, CAUSE_CTICCMSIP420BADEXTENSION, CAUSE_CTICCMSIP421EXTENSTIONREQUIRED, CAUSE_CTICCMSIP423INTERVALTOOBRIEF, CAUSE_CTICCMSIP480TEMPORARILYUNAVAILABLE, CAUSE_CTICCMSIP481CALLLEGDOESNOTEXIST, CAUSE_CTICCMSIP482LOOPDETECTED, CAUSE_CTICCMSIP483TOOMANYHOOPS, CAUSE_CTICCMSIP484ADDRESSINCOMPLETE, CAUSE_CTICCMSIP485AMBIGUOUS, CAUSE_CTICCMSIP486BUSYHERE, CAUSE_CTICCMSIP487REQUESTTERMINATED, CAUSE_CTICCMSIP488NOTACCEPTABLEHERE, CAUSE_CTICCMSIP491REQUESTPENDING, CAUSE_CTICCMSIP493UNDECIPHERABLE, CAUSE_CTICCMSIP500SERVERINTERNALERROR, CAUSE_CTICCMSIP501NOTIMPLEMENTED, CAUSE_CTICCMSIP502BADGATEWAY, CAUSE_CTICCMSIP503SERVICEUNAVAILABLE, CAUSE_CTICCMSIP504SERVERTIMEOUT, CAUSE_CTICCMSIP505SIPVERSIONNOTSUPPORTED, CAUSE_CTICCMSIP513MESSAGETOOLARGE, CAUSE_CTICCMSIP600BUSYEVERYWHERE, CAUSE_CTICCMSIP603DECLINE, CAUSE_CTICCMSIP604DOESNOTEXISTANYWHERE, CAUSE_CTICCMSIP606NOTACCEPTABLE, CAUSE_CTICONFERENCEFULL, CAUSE_CTIDEVICENOTPREEMPTABLE, CAUSE_CTIDROPCONFEREE, CAUSE_CTIMANAGER_FAILURE, CAUSE_CTIPRECEDENCECALLBLOCKED, CAUSE_CTIPRECEDENCELEVELEXCEEDED, CAUSE_CTIPRECEDENCEOUTOFBANDWIDTH, CAUSE_CTIPREEMPTFORREUSE, CAUSE_CTIPREEMPTNOREUSE, CAUSE_DESTINATIONOUTOFORDER, CAUSE_DESTNUMMISSANDDCNOTSUB, CAUSE_DPARK, CAUSE_DPARK_REMINDER, CAUSE_DPARK_UNPARK, CAUSE_EXCHANGEROUTINGERROR, CAUSE_FAC_CMC, CAUSE_FACILITYREJECTED, CAUSE_IDENTIFIEDCHANDOESNOTEXIST, CAUSE_IENIMPL, CAUSE_INBOUNDBLINDTRANSFER, CAUSE_INBOUNDCONFERENCE, CAUSE_INBOUNDTRANSFER, CAUSE_INCOMINGCALLBARRED, CAUSE_INCOMPATABLEDDESTINATION, CAUSE_INTERWORKINGUNSPECIFIED, CAUSE_INVALIDCALLREFVALUE, CAUSE_INVALIDIECONTENTS, CAUSE_INVALIDMESSAGEUNSPECIFIED, CAUSE_INVALIDNUMBERFORMAT, CAUSE_INVALIDTRANSITNETSEL, CAUSE_MANDATORYIEMISSING, CAUSE_MSGNCOMPATABLEWCS, CAUSE_MSGTYPENCOMPATWCS, CAUSE_MSGTYPENIMPL, CAUSE_NETOUTOFORDER, CAUSE_NOANSWERFROMUSER, CAUSE_NOCALLSUSPENDED, CAUSE_NOCIRCAVAIL, CAUSE_NOERROR, CAUSE_NONSELECTEDUSERCLEARING, CAUSE_NORMALCALLCLEARING, CAUSE_NORMALUNSPECIFIED, CAUSE_NOROUTETODDESTINATION, CAUSE_NOROUTETOTRANSITNET, CAUSE_NOUSERRESPONDING, CAUSE_NUMBERCHANGED, CAUSE_ONLYRDIVEARERCAPAVAIL, CAUSE_OUTBOUNDCONFERENCE,

CAUSE_OUTBOUNDTRANSFER, CAUSE_OUTOFBANDWIDTH, CAUSE_PROTOCOLERRORUNSPECIFIED, CAUSE_QSIG_PR, CAUSE_QUALOFSERVNAVAIL, CAUSE_QUIET_CLEAR, CAUSE_RECOVERYONTIMEREXPIRY, CAUSE_REDIRECTED, CAUSE_REQCALLIDHASBEENCLEARED, CAUSE_REQCIRCNAVIL, CAUSE_REQFACILITYNIMPL, CAUSE_REQFACILITYNOTSUBSCRIBED, CAUSE_RESOURCESNAVAIL, CAUSE_RESPONSETOSTATUSENQUIRY, CAUSE_SERVNOTAVAILUNSPECIFIED, CAUSE_SERVOPERATIONVIOLATED, CAUSE_SERVOROPTNAVAILORIMPL, CAUSE_SUBSCRIBERABSENT, CAUSE_SUSPCALLBUTNOTTHISONE, CAUSE_SWITCHINGEQUIPMENTCONGESTION, CAUSE_TEMPORARYFAILURE, CAUSE_UNALLOCATEDNUMBER, CAUSE_USERBUSY

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 75: Methods in CiscoConferenceChainAddedEv

Interface	Method	Description
javax.telephony.Connection	getAddedConnection()	Returns the conference chain Connection that was added to the call.
CiscoConferenceChain	getConferenceChain()	Returns a CiscoConferenceChain that contains all of the conference connections for the calls that are chained together.

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoCallEv

getCiscoCause, getCiscoFeatureReason

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values for more information.

CiscoConferenceChainRemovedEv

The system sends a CiscoConferenceChainRemovedEv event when a conference chain connection gets removed from a call. This event gets sent whenever a conference chain connection gets removed. This event gets reported via theCallControlCallObserver interface.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.events.CallEv, CiscoCallEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoConferenceChainRemovedEv extends CiscoCallEv

Fields

Table 76: Fields in CiscoConferenceChainRemovedEv

Interface	Field
static int	ID

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoCallEv

CAUSE_ACCESSINFORMATIONDISCARDED, CAUSE_BARGE, CAUSE_BCBPRESENTLYAVAIL, CAUSE_BCNAUTHORIZED, CAUSE_BEARERCAPNIMPL, CAUSE_CALLBEINGDELIVERED, CAUSE_CALLIDINUSE, CAUSE_CALLMANAGER_FAILURE, CAUSE_CALLREJECTED, CAUSE_CALLSPLIT, CAUSE_CHANTYPENIMPL, CAUSE_CHANUNACCEPTABLE, CAUSE_CTICCMSIP400BADREQUEST, CAUSE_CTICCMSIP401UNAUTHORIZED, CAUSE_CTICCMSIP402PAYMENTREQUIRED, CAUSE_CTICCMSIP403FORBIDDEN, CAUSE_CTICCMSIP404NOTFOUND, CAUSE_CTICCMSIP405METHODNOTALLOWED, CAUSE_CTICCMSIP406NOTACCEPTABLE, CAUSE_CTICCMSIP407PROXYAUTHENTICATIONREQUIRED, CAUSE_CTICCMSIP408REQUESTTIMEOUT, CAUSE_CTICCMSIP410GONE, CAUSE_CTICCMSIP411LENGTHREQUIRED, CAUSE_CTICCMSIP413REQUESTENTITYTOOLONG, CAUSE_CTICCMSIP414REQUESTURITOOLONG, CAUSE_CTICCMSIP415UNSUPPORTEDMEDIATYPE, CAUSE_CTICCMSIP416UNSUPPORTEDURISCHEME, CAUSE_CTICCMSIP420BADEXTENSION, CAUSE_CTICCMSIP421EXTENSTIONREQUIRED, CAUSE_CTICCMSIP423INTERVALTOOBRIEF, CAUSE_CTICCMSIP480TEMPORARILYUNAVAILABLE, CAUSE_CTICCMSIP481CALLLEGDOESNOTEXIST, CAUSE_CTICCMSIP482LOOPDETECTED, CAUSE_CTICCMSIP483TOOMANYHOOPS, CAUSE_CTICCMSIP484ADDRESSINCOMPLETE, CAUSE_CTICCMSIP485AMBIGUOUS, CAUSE_CTICCMSIP486BUSYHERE, CAUSE_CTICCMSIP487REQUESTTERMINATED, CAUSE_CTICCMSIP488NOTACCEPTABLEHERE, CAUSE_CTICCMSIP491REQUESTPENDING, CAUSE_CTICCMSIP493UNDECIPHERABLE, CAUSE_CTICCMSIP500SERVERINTERNALERROR, CAUSE_CTICCMSIP501NOTIMPLEMENTED, CAUSE_CTICCMSIP502BADGATEWAY, CAUSE_CTICCMSIP503SERVICEUNAVAILABLE, CAUSE_CTICCMSIP504SERVERTIMEOUT, CAUSE_CTICCMSIP505SIPVERSIONNOTSUPPORTED, CAUSE_CTICCMSIP513MESSAGETOOLARGE, CAUSE_CTICCMSIP600BUSYEVERYWHERE, CAUSE_CTICCMSIP603DECLINE, CAUSE_CTICCMSIP604DOESNOTEXISTANYWHERE, CAUSE_CTICCMSIP606NOTACCEPTABLE, CAUSE_CTICONFERENCEFULL, CAUSE_CTIDEVICENOTPREEMPTABLE, CAUSE_CTIDROPCONFEREE, CAUSE_CTIMANAGER_FAILURE, CAUSE_CTIPRECEDENCECALLBLOCKED, CAUSE_CTIPRECEDENCELEVELEXCEEDED, CAUSE_CTIPRECEDENCEOUTOFBANDWIDTH, CAUSE_CTIPREEMPTFORREUSE, CAUSE_CTIPREEMPTNOREUSE, CAUSE_DESTINATIONOUTOFORDER, CAUSE_DESTNUMMISSANDDCNOTSUB, CAUSE_DPARK, CAUSE_DPARK_REMINDER, CAUSE_DPARK_UNPARK, CAUSE_EXCHANGEROUTINGERROR, CAUSE_FAC_CMC, CAUSE_FACILITYREJECTED, CAUSE_IDENTIFIEDCHANDOESNOTEXIST, CAUSE_IENIMPL, CAUSE_INBOUNDBLINDTRANSFER, CAUSE_INBOUNDCONFERENCE, CAUSE_INBOUNDTRANSFER, CAUSE_INCOMINGCALLBARRED, CAUSE_INCOMPATABLEDDESTINATION, CAUSE_INTERWORKINGUNSPECIFIED, CAUSE_INVALIDCALLREFVALUE, CAUSE_INVALIDIECONTENTS, CAUSE_INVALIDMESSAGEUNSPECIFIED, CAUSE_INVALIDNUMBERFORMAT, CAUSE_INVALIDTRANSITNETSEL, CAUSE_MANDATORYIEMISSING, CAUSE_MSGNCOMPATABLEWCS, CAUSE_MSGTYPENCOMPATWCS, CAUSE_MSGTYPENIMPL, CAUSE_NETOUTOFORDER, CAUSE_NOANSWERFROMUSER, CAUSE_NOCALLSUSPENDED, CAUSE_NOCIRCAVAIL, CAUSE_NOERROR, CAUSE_NONSELECTEDUSERCLEARING, CAUSE_NORMALCALLCLEARING, CAUSE_NORMALUNSPECIFIED, CAUSE_NOROUTETODDESTINATION, CAUSE_NOROUTETOTRANSITNET, CAUSE_NOUSERRESPONDING, CAUSE_NUMBERCHANGED, CAUSE_ONLYRDIVEARERCAPAVAIL, CAUSE_OUTBOUNDCONFERENCE,

CAUSE_OUTBOUNDTRANSFER, CAUSE_OUTOFBANDWIDTH, CAUSE_PROTOCOLERRORUNSPECIFIED, CAUSE_QSIG_PR, CAUSE_QUALOFSERVNAVAIL, CAUSE_QUIET_CLEAR, CAUSE_RECOVERYONTIMEREXPIRY, CAUSE_REDIRECTED, CAUSE_REQCALLIDHASBEENCLEARED, CAUSE_REQCIRCNAVIL, CAUSE_REQFACILITYNIMPL, CAUSE_REQFACILITYNOTSUBSCRIBED, CAUSE_RESOURCESNAVAIL, CAUSE_RESPONSETOSTATUSENQUIRY, CAUSE_SERVNOTAVAILUNSPECIFIED, CAUSE_SERVOPERATIONVIOLATED, CAUSE_SERVOROPTNAVAILORIMPL, CAUSE_SUBSCRIBERABSENT, CAUSE_SUSPCALLBUTNOTTHISONE, CAUSE_SWITCHINGEQUIPMENTCONGESTION, CAUSE_TEMPORARYFAILURE, CAUSE_UNALLOCATEDNUMBER, CAUSE_USERBUSY

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 77: Methods in CiscoConferenceChainRemovedEf

Interface	Method	Description
CiscoConferenceChain	getConferenceChain()	Returns a CiscoConferenceChain that contains all of the conference connections for the calls that are chained together. Returns: Connection.
javax.telephony.Connection	getRemovedConnection()	Returns the conference chain Connection that was removed from the call. Returns: CiscoConferenceChain.

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoCallEv

getCiscoCause, getCiscoFeatureReason

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values for more information.

CiscoConferenceEndEv

The CiscoConferenceEndEv event indicates that a Conference operation completed. The system reports this event via the CallControlCallObserver interface.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1)	Created history table to track changes.

Superinterfaces

javax.telephony.events.CallEv, CiscoCallEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoConferenceEndEv extends CiscoCallEv

Fields

Table 78: Fields in CiscoConferenceEndEv

Interface	Field
static int	ID

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoCallEv

CAUSE_ACCESSINFORMATIONDISCARDED, CAUSE_BARGE, CAUSE_BCBPRESENTLYAVAIL, CAUSE_BCNAUTHORIZED, CAUSE_BEARERCAPNIMPL, CAUSE_CALLBEINGDELIVERED, CAUSE_CALLIDINUSE, CAUSE_CALLMANAGER_FAILURE, CAUSE_CALLREJECTED, CAUSE_CALLSPLIT, CAUSE_CHANTYPENIMPL, CAUSE_CHANUNACCEPTABLE, CAUSE_CTICCMSIP400BADREQUEST, CAUSE_CTICCMSIP401UNAUTHORIZED, CAUSE_CTICCMSIP402PAYMENTREQUIRED, CAUSE_CTICCMSIP403FORBIDDEN, CAUSE_CTICCMSIP404NOTFOUND, CAUSE_CTICCMSIP405METHODNOTALLOWED, CAUSE_CTICCMSIP406NOTACCEPTABLE, CAUSE_CTICCMSIP407PROXYAUTHENTICATIONREQUIRED, CAUSE_CTICCMSIP408REQUESTTIMEOUT, CAUSE_CTICCMSIP410GONE, CAUSE_CTICCMSIP411LENGTHREQUIRED, CAUSE_CTICCMSIP413REQUESTENTITYTOOLONG, CAUSE_CTICCMSIP414REQUESTURITOOLONG, CAUSE_CTICCMSIP415UNSUPPORTEDMEDIATYPE, CAUSE_CTICCMSIP416UNSUPPORTEDURISCHEME, CAUSE_CTICCMSIP420BADEXTENSION, CAUSE_CTICCMSIP421EXTENSTIONREQUIRED, CAUSE_CTICCMSIP423INTERVALTOOBRIEF, CAUSE_CTICCMSIP480TEMPORARILYUNAVAILABLE, CAUSE_CTICCMSIP481CALLLEGDOESNOTEXIST, CAUSE_CTICCMSIP482LOOPDETECTED, CAUSE_CTICCMSIP483TOOMANYHOOPS, CAUSE_CTICCMSIP484ADDRESSINCOMPLETE, CAUSE_CTICCMSIP485AMBIGUOUS, CAUSE_CTICCMSIP486BUSYHERE, CAUSE_CTICCMSIP487REQUESTTERMINATED, CAUSE_CTICCMSIP488NOTACCEPTABLEHERE, CAUSE_CTICCMSIP491REQUESTPENDING, CAUSE_CTICCMSIP493UNDECIPHERABLE, CAUSE_CTICCMSIP500SERVERINTERNALERROR, CAUSE_CTICCMSIP501NOTIMPLEMENTED, CAUSE_CTICCMSIP502BADGATEWAY, CAUSE_CTICCMSIP503SERVICEUNAVAILABLE, CAUSE_CTICCMSIP504SERVERTIMEOUT, CAUSE_CTICCMSIP505SIPVERSIONNOTSUPPORTED, CAUSE_CTICCMSIP513MESSAGETOOLARGE, CAUSE_CTICCMSIP600BUSYEVERYWHERE, CAUSE_CTICCMSIP603DECLINE, CAUSE_CTICCMSIP604DOESNOTEXISTANYWHERE, CAUSE_CTICCMSIP606NOTACCEPTABLE, CAUSE_CTICONFERENCEFULL, CAUSE_CTIDEVICENOTPREEMPTABLE, CAUSE_CTIDROPCONFEREE, CAUSE_CTIMANAGER_FAILURE, CAUSE_CTIPRECEDENCECALLBLOCKED, CAUSE_CTIPRECEDENCELEVELEXCEEDED, CAUSE_CTIPRECEDENCEOUTOFBANDWIDTH, CAUSE_CTIPREEMPTFORREUSE, CAUSE_CTIPREEMPTNOREUSE, CAUSE_DESTINATIONOUTOFORDER, CAUSE_DESTNUMMISSANDDCNOTSUB, CAUSE_DPARK, CAUSE_DPARK_REMINDER, CAUSE_DPARK_UNPARK, CAUSE_EXCHANGEROUTINGERROR, CAUSE_FAC_CMC, CAUSE_FACILITYREJECTED, CAUSE_IDENTIFIEDCHANDOESNOTEXIST, CAUSE_IENIMPL, CAUSE_INBOUNDBLINDTRANSFER, CAUSE_INBOUNDCONFERENCE, CAUSE_INBOUNDTRANSFER, CAUSE_INCOMINGCALLBARRED, CAUSE_INCOMPATABLEDDESTINATION, CAUSE_INTERWORKINGUNSPECIFIED, CAUSE_INVALIDCALLREFVALUE, CAUSE_INVALIDIECONTENTS, CAUSE_INVALIDMESSAGEUNSPECIFIED, CAUSE_INVALIDNUMBERFORMAT, CAUSE_INVALIDTRANSITNETSEL, CAUSE_MANDATORYIEMISSING, CAUSE_MSGNCOMPATABLEWCS, CAUSE_MSGTYPENCOMPATWCS, CAUSE_MSGTYPENIMPL, CAUSE_NETOUTOFORDER, CAUSE_NOANSWERFROMUSER, CAUSE_NOCALLSUSPENDED, CAUSE_NOCIRCAVAIL, CAUSE_NOERROR, CAUSE_NONSELECTEDUSERCLEARING, CAUSE_NORMALCALLCLEARING, CAUSE_NORMALUNSPECIFIED, CAUSE_NOROUTETODDESTINATION, CAUSE_NOROUTETOTRANSITNET, CAUSE_NOUSERRESPONDING, CAUSE_NUMBERCHANGED, CAUSE_ONLYRDIVEARERCAPAVAIL, CAUSE_OUTBOUNDCONFERENCE,

CAUSE_OUTBOUNDTRANSFER, CAUSE_OUTOFBANDWIDTH, CAUSE_PROTOCOLERRORUNSPECIFIED, CAUSE_QSIG_PR, CAUSE_QUALOFSERVNAVAIL, CAUSE_QUIET_CLEAR, CAUSE_RECOVERYONTIMEREXPIRY, CAUSE_REDIRECTED, CAUSE_REQCALLIDHASBEENCLEARED, CAUSE_REQCIRCNAVIL, CAUSE_REQFACILITYNIMPL, CAUSE_REQFACILITYNOTSUBSCRIBED, CAUSE_RESOURCESNAVAIL, CAUSE_RESPONSETOSTATUSENQUIRY, CAUSE_SERVNOTAVAILUNSPECIFIED, CAUSE_SERVOPERATIONVIOLATED, CAUSE_SERVOROPTNAVAILORIMPL, CAUSE_SUBSCRIBERABSENT, CAUSE_SUSPCALLBUTNOTTHISONE, CAUSE_SWITCHINGEQUIPMENTCONGESTION, CAUSE_TEMPORARYFAILURE, CAUSE_UNALLOCATEDNUMBER, CAUSE_USERBUSY

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 79: Methods in CiscoConferenceEndEv

Interface	Method	Description
javax.telephony.Address	getConferenceControllerAddress()	Returns the Address that currently acts as the conference controller for this call, the initiating call.
javax.telephony.Call	getConferencedCall()	Returns the call that merged. This call is in the Call.INVALID state.
javax.telephony.Call[]	getFailedCalls()	Returns list of Calls that could not be Conferenced.
javax.telephony.Call	getFinalCall()	Returns the call that remains active after the conference completes.

Interface	Method	Description
javax.telephony.TerminalConnection	getHeldConferenceController()	Returns the TerminalConnection that currently acts as the conference controller for this call -- the final call. This is the TerminalConnection that was in HELD state when the conference got initiated. This method returns null or TerminalConnection if the conference controller is not being observed.
javax.telephony.TerminalConnection	getTalkingConferenceController()	Returns the TerminalConnection that currently acts as the conference controller for this call -- the initiating call.This is the TerminalConnection that was in TALKING state. This method returns null or TerminalConnection if the conference controller is not being observed.
boolean	isSuccess()	Returns Boolean True or False depending on whether the conference succeeded or failed. The application can use this interface to determine whether a Conference is successful. Conferences will fail in these situations: • If the application issues the request Call.conference(otherCalls[]), the system considers the conference as failed if one or more than one Calls could not Join into Conference. Use getFailedCalls() to find the failed calls. • If no conference bridge is available, and the conference could not complete. Use getFailedCalls() to get a list of the calls that could not join the conference. • If the party being conferenced drops out before the conference could complete.

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoCallEv

getCiscoCause, getCiscoFeatureReason

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values

See also isSuccess()

CiscoConferenceStartEv

The CiscoConferenceStartEv event indicates that a conference operation started. The CallControlCallObserver interface reports this event.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Added getControllerTerminalName() method for Join Across Lines/Connected Conference feature.

Superinterfaces

javax.telephony.events.CallEv, CiscoCallEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoConferenceStartEv extends CiscoCallEv

Fields

Table 80: Fields in CiscoConferenceStartEv

Interface	Field
static int	ID

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoCallEv

CAUSE_ACCESSINFORMATIONDISCARDED, CAUSE_BARGE, CAUSE_BCBPRESENTLYAVAIL, CAUSE_BCNAUTHORIZED, CAUSE_BEARERCAPNIMPL, CAUSE_CALLBEINGDELIVERED, CAUSE_CALLIDINUSE, CAUSE_CALLMANAGER_FAILURE, CAUSE_CALLREJECTED, CAUSE_CALLSPLIT, CAUSE_CHANTYPENIMPL, CAUSE_CHANUNACCEPTABLE, CAUSE_CTICCMSIP400BADREQUEST, CAUSE_CTICCMSIP401UNAUTHORIZED, CAUSE_CTICCMSIP402PAYMENTREQUIRED, CAUSE_CTICCMSIP403FORBIDDEN, CAUSE_CTICCMSIP404NOTFOUND, CAUSE_CTICCMSIP405METHODNOTALLOWED, CAUSE_CTICCMSIP406NOTACCEPTABLE, CAUSE_CTICCMSIP407PROXYAUTHENTICATIONREQUIRED, CAUSE_CTICCMSIP408REQUESTTIMEOUT, CAUSE_CTICCMSIP410GONE, CAUSE_CTICCMSIP411LENGTHREQUIRED, CAUSE_CTICCMSIP413REQUESTENTITYTOOLONG, CAUSE_CTICCMSIP414REQUESTURITOOLONG, CAUSE_CTICCMSIP415UNSUPPORTEDMEDIATYPE, CAUSE_CTICCMSIP416UNSUPPORTEDURISCHEME, CAUSE_CTICCMSIP420BADEXTENSION, CAUSE_CTICCMSIP421EXTENSTIONREQUIRED, CAUSE_CTICCMSIP423INTERVALTOOBRIEF, CAUSE_CTICCMSIP480TEMPORARILYUNAVAILABLE, CAUSE_CTICCMSIP481CALLLEGDOESNOTEXIST, CAUSE_CTICCMSIP482LOOPDETECTED, CAUSE_CTICCMSIP483TOOMANYHOOPS, CAUSE_CTICCMSIP484ADDRESSINCOMPLETE, CAUSE_CTICCMSIP485AMBIGUOUS, CAUSE_CTICCMSIP486BUSYHERE, CAUSE_CTICCMSIP487REQUESTTERMINATED, CAUSE_CTICCMSIP488NOTACCEPTABLEHERE, CAUSE_CTICCMSIP491REQUESTPENDING, CAUSE_CTICCMSIP493UNDECIPHERABLE, CAUSE_CTICCMSIP500SERVERINTERNALERROR, CAUSE_CTICCMSIP501NOTIMPLEMENTED, CAUSE_CTICCMSIP502BADGATEWAY, CAUSE_CTICCMSIP503SERVICEUNAVAILABLE, CAUSE_CTICCMSIP504SERVERTIMEOUT, CAUSE_CTICCMSIP505SIPVERSIONNOTSUPPORTED, CAUSE_CTICCMSIP513MESSAGETOOLARGE, CAUSE_CTICCMSIP600BUSYEVERYWHERE, CAUSE_CTICCMSIP603DECLINE, CAUSE_CTICCMSIP604DOESNOTEXISTANYWHERE, CAUSE_CTICCMSIP606NOTACCEPTABLE, CAUSE_CTICONFERENCEFULL, CAUSE_CTIDEVICENOTPREEMPTABLE, CAUSE_CTIDROPCONFEREE, CAUSE_CTIMANAGER_FAILURE, CAUSE_CTIPRECEDENCECALLBLOCKED, CAUSE_CTIPRECEDENCELEVELEXCEEDED, CAUSE_CTIPRECEDENCEOUTOFBANDWIDTH, CAUSE_CTIPREEMPTFORREUSE, CAUSE_CTIPREEMPTNOREUSE, CAUSE_DESTINATIONOUTOFORDER, CAUSE_DESTNUMMISSANDDCNOTSUB, CAUSE_DPARK, CAUSE_DPARK_REMINDER, CAUSE_DPARK_UNPARK, CAUSE_EXCHANGEROUTINGERROR, CAUSE_FAC_CMC, CAUSE_FACILITYREJECTED, CAUSE_IDENTIFIEDCHANDOESNOTEXIST, CAUSE_IENIMPL, CAUSE_INBOUNDBLINDTRANSFER, CAUSE_INBOUNDCONFERENCE, CAUSE_INBOUNDTRANSFER, CAUSE_INCOMINGCALLBARRED, CAUSE_INCOMPATABLEDDESTINATION, CAUSE_INTERWORKINGUNSPECIFIED, CAUSE_INVALIDCALLREFVALUE, CAUSE_INVALIDIECONTENTS, CAUSE_INVALIDMESSAGEUNSPECIFIED, CAUSE_INVALIDNUMBERFORMAT, CAUSE_INVALIDTRANSITNETSEL, CAUSE_MANDATORYIEMISSING, CAUSE_MSGNCOMPATABLEWCS, CAUSE_MSGTYPENCOMPATWCS, CAUSE_MSGTYPENIMPL, CAUSE_NETOUTOFORDER, CAUSE_NOANSWERFROMUSER, CAUSE_NOCALLSUSPENDED, CAUSE_NOCIRCAVAIL, CAUSE_NOERROR, CAUSE_NONSELECTEDUSERCLEARING, CAUSE_NORMALCALLCLEARING, CAUSE_NORMALUNSPECIFIED, CAUSE_NOROUTETODDESTINATION, CAUSE_NOROUTETOTRANSITNET, CAUSE_NOUSERRESPONDING, CAUSE_NUMBERCHANGED, CAUSE_ONLYRDIVEARERCAPAVAIL, CAUSE_OUTBOUNDCONFERENCE,

CAUSE_OUTBOUNDTRANSFER, CAUSE_OUTOFBANDWIDTH, CAUSE_PROTOCOLERRORUNSPECIFIED, CAUSE_QSIG_PR, CAUSE_QUALOFSERVNAVAIL, CAUSE_QUIET_CLEAR, CAUSE_RECOVERYONTIMEREXPIRY, CAUSE_REDIRECTED, CAUSE_REQCALLIDHASBEENCLEARED, CAUSE_REQCIRCNAVIL, CAUSE_REQFACILITYNIMPL, CAUSE_REQFACILITYNOTSUBSCRIBED, CAUSE_RESOURCESNAVAIL, CAUSE_RESPONSETOSTATUSENQUIRY, CAUSE_SERVNOTAVAILUNSPECIFIED, CAUSE_SERVOPERATIONVIOLATED, CAUSE_SERVOROPTNAVAILORIMPL, CAUSE_SUBSCRIBERABSENT, CAUSE_SUSPCALLBUTNOTTHISONE, CAUSE_SWITCHINGEQUIPMENTCONGESTION, CAUSE_TEMPORARYFAILURE, CAUSE_UNALLOCATEDNUMBER, CAUSE_USERBUSY

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 81: Methods in CiscoConferenceStartEv

Interface	Method	Description
javax.telephony.Address	getConferenceControllerAddress()	Returns the Address that currently acts as the conference controller for this call, the initiating call.
javax.telephony.Call	getConferencedCall()	Returns the call that will be conferenced. This is the call that will be merged into the initiating call. This interface returns the first call from the list of calls that are joining into conference.
javax.telephony.Call[]	getConferencedCalls()	Returns the list of the calls that will be conferenced. These calls are the ones that will be merged into the final call.

Interface	Method	Description
javax.telephony.Call	getFinalCall()	Returns the call that will remain active after the conference completes. This is the call into which all the calls will merge.
javax.telephony.TerminalConnection	getHeldConferenceController()	Returns the TerminalConnection that currently acts as the conference controller for this call, the initiating call. This is the TerminalConnection that was in HELD state. This method returns null if the conference controller is not being observed. This method returns the first held controller for a multiple call join scenario.
javax.telephony.TerminalConnection[]	getHeldConferenceControllers()	Returns all TerminalConnections on Conference Controller Terminal that are joining together and are in HELD State.
javax.telephony.Address	getOriginalConferenceControllerAddress()	Returns the Address of the participant that initiated the conference.
javax.telephony.TerminalConnection	getTalkingConferenceController()	Returns the TerminalConnection that currently acts as the conference controller for this call, the initiating call. This is the TerminalConnection that was in TALKING state. This method returns null if the conference controller is not being observed. This method returns null if there is no controller in talking state. Calls can be joined into a conference without any talking controller.
String	getControllerTerminalName()	Returns the terminal name of the controllers across which a conference is done.

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoCallEv

getCiscoCause, getCiscoFeatureReason

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoConnection

The CiscoConnection interface extends the CallControlConnection interface with additional Cisco specific capabilities. Applications can use the getReason method to obtain the reason for the creation of a connection.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Added two new methods: getPartyInfo and disconnect(CiscoPartyInfopartyInfo for Drop Any Party feature.
8.0(1)	Enhanced with the following: • New method to get the associated CiscoHuntConnection. If application is observing hunt list member, applications can use this method to find out if call is routed through HuntPilot. • New interface getUniqueID(Terminal term) is added which will return the uniqueID as string. • New method that allows an application to determine if the connection is associated with a chaperone device on a chaperone call. Chapone devices have a limited feature set, and knowing that a connection is associated with a chaperone device can allow the application to better handle the connections.
11.5(1)	Added redirect method with deviceName.

All Superinterfaces

javax.telephony.callcontrol.CallControlConnection, CiscoObjectContainer, javax.telephony.Connection

Declaration

public interface CiscoConnection extends javax.telephony.callcontrol.CallControlConnection, CiscoObjectContainer

Fields

Table 82: Fields in CiscoConnection

Interface	Field	Description
static int	ADDRESS_SEARCH_SPACE	The redirect should be done by using the redirect controller address search space.
static int	CALLED_ADDRESS_DEFAULT	The default behavior for Cisco JTAPI should apply.
static int	CALLED_ADDRESS_SET_TO_PREFERRED CALLEDPARTY	The original called Address should be set to the value present in preferredOriginalCalledParty field.
static int	CALLED_ADDRESS_SET_TO_REDIRECT_ DESTINATION	The called Address should be reset to the redirect destination.
static int	CALLED_ADDRESS_UNCHANGED	The called Address should remain unchanged after the redirect operation.
static int	CALLINGADDRESS_SEARCH_SPACE	The redirect should be done by using the calling address search space.
static int	DEFAULT_SEARCH_SPACE	The redirect should be done by using the default search space for the implementation.
static int	REASON_DIRECTCALL	This Connection results from a direct call.
static int	REASON_FORWARDALL	This Connection results from unconditional forwarding.
static int	REASON_FORWARDBUSY	This Connection results from a forwarding on busy.
static int	REASON_FORWARDNOANSWER	This Connection results from a forwarding on no answer.
static int	REASON_OUTBOUND	This Connection is an originating Connection, not a destination Connection.
static int	REASON_REDIRECT	This Connection results from a redirection.

Interface	Field	Description
static int	REASON_TRANSFERREDCALL	This Connection results from a transfer.
static int	REDIRECT_DROP_ON_FAILURE	This redirect mode instructs the implementation to perform redirect without checking the validity or availability of the destination.
static int	REDIRECT_NORMAL	This redirect mode instructs the implementation to perform redirect if the destination is valid and available.

Inherited Fields

From Interface javax.telephony.callcontrol.CallControlConnection

ALERTING, DIALING, DISCONNECTED, ESTABLISHED, FAILED, IDLE, INITIATED, NETWORK_ALERTING, NETWORK_REACHED, OFFERED, OFFERING, QUEUED, UNKNOWN

From Interface javax.telephony.Connection

CONNECTED, INPROGRESS

Methods

Table 83: Methods in CiscoConnection

Interface	Method and Descrption
Boolean	getAddressPI()
	Returns Presentation Indicator (PI) associated with the Address on which the connection is created.
CiscoHuntConnection	getCiscoHuntConnection()
	This method returns the associated CiscoHuntConnection or null.
CiscoConnectionID	getConnectionID()
	Returns CiscoConnectionID for this CiscoConnection
java.lang.String	getDParkPrefixCode()
	Returns the prefix code that needs to be dialed with the DPark DN to retrieve the call.

Interface	Method and Descrption
int	getReason()
	Returns the reason for the creation of this Connection. To function properly, some applications need to know the reason for the creation of the connection is created at an endpoint. The reason for a Connection creation may be any of the following constants: • CiscoConnection. REASON_DIRECTCALL CiscoConnection. REASON_TRANSFERREDCALL • CiscoConnection. REASON_FORWARDNOANSWER • CiscoConnection. REASON_FORWARDBUSY • CiscoConnection. REASON_FORWARDALL • CiscoConnection. REASON_REDIRECT • CiscoConnection. REASON_NORMAL
javax.telephony.TerminalConnection	getRequestController()
	Returns the current request Controller for the Connection.
String	getUniqueID(Terminal term)
	This method returns the updated uniqueID of the connection. In case if there are no shared lines associated with this connection, application can just pass null object as parameter to this interface to get the Unique Identifier. Unique Identifier will be same for all the shared lines, but if it's a barge scenario, different terminals would have different identifier.The returned Unique Identifier will be 32-character hex string. Please refer to End to End Call TracingEnd to End Call Tracing, for more information. **Throws:** PrivilegeVoilationException , InvalidStateException **Parameters:** Terminal
boolean	isChaperone()
	This method returns true if the connection is associated with a Chaperone call, and false if not.
java.lang.String	park()
	This method parks the call at a system park DN and returns the address of the park DN.

Interface	Method and Descrption
javax.telephony.Connection	redirect(java.lang.String destinationAddress, int mode)
	This method overloads the CallControlConnection.redirect() method. **Throws** javax.telephony. InvalidStateException javax.telephony. InvalidPartyException javax.telephony. MethodNotSupportedException javax.telephony. PrivilegeViolationException javax.telephony. ResourceUnavailableException **Parameter** Mode - This parameter can take one of the following two values: • CiscoConnection. REDIRECT_DROP_ON_FAILURE: This mode instructs the implementation to perform a redirect without checking the validity or availability of the destination. The original call gets dropped if the destination is invalid or busy. • CiscoConnection. REDIRECT_NORMAL: This mode instructs the implementation to perform a redirect only after checking the validity or availability of the destination. This matches the behavior of the CallControlConnection.redirect() method. The system does not drop the original call on failure.

Interface	Method and Descrption
javax.telephony.Connection	redirect(java.lang.String destinationAddress, int mode, int callingSearchSpace)
	This method overloads the CallControlConnection.redirect() method. It takes two new parameters: redirectMode and callingSearchSpace. The redirectMode selects which type of redirect to perform. The callingSearchSpace tells the implementation to use either the calling party search space or the redirect controller search space. **Parameters** mode - One of the following: • CiscoConnection.REDIRECT_DROP_ON_FAILURE: This mode instructs the implementation to perform a redirect without checking the validity or availability of the destination. The original call gets dropped if the destination is invalid or busy. • CiscoConnection.REDIRECT_NORMAL: This mode instructs the implementation to perform a redirect only after checking the validity or availability of the destination. This matches the behavior of the CallControlConnection.redirect() method. The system does not drop the original call on failure. callingSearchSpace - One of the following: • CiscoConnection. DEFAULT_SEARCH_SPACE • CiscoConnection. CALLINGADDRESS_SEARCH_SPACE • CiscoConnection. ADDRESS_SEARCH_SPACE **Throws** javax.telephony. InvalidStateException javax.telephony. InvalidPartyException javax.telephony. MethodNotSupportedException javax.telephony. PrivilegeViolationException javax.telephony. ResourceUnavailableException

Interface	Method and Descrption
javax.telephony.Connection	redirect(java.lang.String destinationAddress, int mode, int callingSearchSpace, int calledAddressOption)
	This method overloads the CallControlConnection.redirect() method. It takes three new parameters: mode, callingSearchSpace, and calledAddressOption. The redirectMode selects the type of redirect to perform. The callingSearchSpace tells the implementation to use either the calling party search space or the redirect controller search space. The calledAddressOption parameter controls whether to reset the original called fields. **Parameters** mode - One of the following: • CiscoConnection. REDIRECT_DROP_ON_FAILURE • CiscoConnection. REDIRECT_NORMAL callingSearchSpace - One of the following: • CiscoConnection. DEFAULT_SEARCH_SPACE • CiscoConnection. CALLINGADDRESS_SEARCH_SPACE • CiscoConnection. ADDRESS_SEARCH_SPACE calledAddressOption: One of the following: • CiscoConnection. CALLED_ADDRESS_DEFAULT • CiscoConnection. CALLED_ADDRESS_UNCHANGED • CiscoConnection. CALLED_ADDRESS_SET_TO_REDIRECT_DESTINATION **Throws** javax.telephony. InvalidStateException javax.telephony. InvalidPartyException javax.telephony. MethodNotSupportedException javax.telephony. PrivilegeViolationException javax.telephony. ResourceUnavailableException

Interface	Method and Descrption
javax.telephony.Connection	redirect(java.lang.String destinationAddress, int mode, int callingSearchSpace, int calledAddressOption, java.lang.String preferredOriginalCalledParty, java.lang.String facCode, java.lang.String cmcCode)

Interface	Method and Descrption
	This method overloads the CallControlConnection.redirect() method. It takes three new parameters: mode, callingSearchSpace, and calledAddressOption. The redirectMode selects the type of redirect to perform. The callingSearchSpace tells the implementation to use either the calling party search space or the redirect controller search space. The calledAddressOption parameter controls whether to reset the original called fields. If the FAC and CMC codes are missing or invalid, the call might not get offered and platformException may contain one of the following error codes: • CiscoJTAPIException. CTIERR_FAC_CMC _REASON_FAC_NEEDED • CiscoJTAPIException. CTIERR_FAC_CMC _REASON_CMC_NEEDED • CiscoJTAPIException. CTIERR_FAC_CMC _REASON_FAC_CMC_NEEDED • CiscoJTAPIException. CTIERR_FAC_CMC _REASON_FAC_INVALID • CiscoJTAPIException. CTIERR_FAC_CMC _REASON_CMC_INVALID **Parameters** mode - One of the following: • CiscoConnection.REDIRECT_DROP_ON_FAILURE • CiscoConnection. REDIRECT_NORMAL callingSearchSpace - One of the following: • CiscoConnection. DEFAULT_SEARCH_SPACE • CiscoConnection. CALLINGADDRESS_SEARCH_SPACE • CiscoConnection. ADDRESS_SEARCH_SPACE • preferredOriginalCalledParty - may be a DN that will be the originalCalledParty field when call is offered to destinationAddress. If this field * needs to be used, applications must set calledAddressOption as CALLED_ADDRESS_SET_TO_PREFERREDCALLEDPARTY. If applications are not interested in this field, you must pass the default value of null. calledAddressOption - One of the following: • CiscoConnection. CALLED_ADDRESS_DEFAULT • CiscoConnection. CALLED_ADDRESS_UNCHANGED • CiscoConnection. CALLED_ADDRESS_SET_TO_REDIRECT_DESTINATION preferredOriginalCalledParty - may be a DN that will be the originalCalledParty field when call is offered to destinationAddress. If this field * needs to be used, applications must set calledAddressOption as CALLED_ADDRESS_SET_TO_PREFERREDCALLEDPARTY. If applications are not interested in this field, you must pass the default value of null. facCode - required if the destinationAddress requires a forced authorization

Interface	Method and Descrption
	code to offer the call. Pass the FAC in this parameter. Pass the default value of null if the destinationAddress does not require a FAC code. cmcCode - required if the destinationAddress requires a client matter code to offer the call. Pass the CMC in this parameter. Pass the default value of null if the destinationAddress does not require a CMC code.
javax.telephony.Connection	redirect(java.lang.String destinationAddress, int mode, int callingSearchSpace, int calledAddressOption, java.lang.String preferredOriginalCalledParty, java.lang.String facCode, java.lang.String cmcCode, int featurePriority)
	This method overloads CallControlConnection.redirect(). It takes a new parameter, featurePriority, that sets the call priority. The featurePriority parameter may be: • CiscoCall.FEATUREPRIORITY_NORMAL • CiscoCall.FEATUREPRIORITY_URGENT • CiscoCall.FEATUREPRIORITY_EMERGENCY **Returns** Connection **Throws** javax.telephony.InvalidStateException javax.telephony.InvalidPartyException javax.telephony.MethodNotSupportedException javax.telephony.PrivilegeViolationException javax.telephony.ResourceUnavailableException

Interface	Method and Descrption
javax.telephony.Connection	redirect(java.lang.String destinationAddress, int mode, int callingSearchSpace, java.lang.String preferredOriginalCalledParty)
	This method overloads the CallControlConnection.redirect() method. It takes three new parameters: mode, callingSearchSpace, and preferredOriginalCalledParty. The redirectMode selects the type of redirect to perform. The callingSearchSpace tells the implementation to use either the calling party search space or the redirect controller search space. **Parameters** mode - One of the following: • CiscoConnection.REDIRECT_DROP_ON_FAILURE • CiscoConnection.REDIRECT_NORMAL callingSearchSpace - One of the following: • CiscoConnection.DEFAULT_SEARCH_SPACE • CiscoConnection.CALLINGADDRESS_SEARCH_SPACE • CiscoConnection.ADDRESS_SEARCH_SPACE preferredOriginalCalledParty - May be a DN that will be the originalCalledParty field when the call gets offered to the destinationAddress. **Throws** javax.telephony.InvalidStateException javax.telephony.InvalidPartyException javax.telephony.MethodNotSupportedException javax.telephony.PrivilegeViolationException javax.telephony.ResourceUnavailableException

Interface	Method and Descrption
javax.telephony.Connection	redirect(String destinationAddress, int mode, int callingSearchSpace, int calledAddressOption, String preferredOriginalCalledParty, String facCode, String cmcCode, int featurePriority, byte[] applicationXMLData)
	This method is similar to the existing redirect method on the CiscoConnection object, except this one takes an additional parameter, applicationXMLData. Parameters applicationXMLData This parameter was added, and it allows an application to send message header point like SIP contact header info to the receiving end point. The parameter takes xml format as mentioned below. <data> <item> <type>contact</type> <operation>append</operation> <protocol>SIP</protocol> <value>;+sip.instance = "<urn:uuid = *guid*>"</value> </item> </data> **Note** This version only supports: contact, operation: append, protocol: SIP. It can be enhanced to support other protocols and operations in the future. If applications are not interested in this field, you must pass the default value of null.
javax.telephony.Connection	redirect(String destinationAddress, int mode, int callingSearchSpace, int calledAddressOption, String preferredOriginalCalledParty, String facCode, String cmcCode, int featurePriority, byte[] applicationXMLData, String deviceName)
	This method is similar to the above method, but it adds the deviceName parameter, which allows you to send the redirect to a specific device. Even in situations where the target device shares a line with another device, the redirected call goes only to the target device and not to the other device that shares the phone line.
void	setRequestController(javax.telephony.TerminalConnection tc)
	This interface gets provided to a requesting TerminalConnection.
com.cisco.jtapi.extensions.CiscoPartyInfo[]	getPartyInfo()
	Returns a list of participants.

Interface	Method and Descrption
java.lang.void	disconnect(CiscoPartyInfopartyInfo)
	Disconnects participant with whose CiscoPartyInfo matches the passed parameter value; throws exception otherwise. **Throws** PrivilegeViolationException InvalidStateException
getLocalUUID(TerminalConnection termConn)	This method takes a Terminal connection object of the connection and returns the Local Universal Unique Identifier of the party associated with both connection and the terminal connection.
getPeerUUID(TerminalConnection termConn)	This method takes a Terminal connection object of the connection and returns the Local Universal Unique Identifier of the party on the other side of the call. It is a part of both connection and the terminal connection.

Inherited Methods

From Interface javax.telephony.callcontrol.CallControlConnection

accept, addToAddress, getCallControlState, park, redirect, reject

From Interface javax.telephony.Connection

disconnect, getAddress, getCall, getCapabilities, getConnectionCapabilities, getState, getTerminalConnections

From Interface com.cisco.jtapi.extensions.CiscoObjectContainer

getObject, setObject

Documentation

None

CiscoConnectionID

The CiscoConnectionID object represents a unique object that is associated with each connection. Applications may use the object itself or the integer representation of the object that the intValue() method returns.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoObjectContainer

Declaration

public interface CiscoConnectionID extends CiscoObjectContainer

Fields

None

Methods

Table 84: Methods in CiscoConnectionID

Interface	Method	Description
CiscoConnection	getConnection()	Returns the CiscoConnection for the CiscoConnectionID.
Int	intValue()	Returns an integer representation of this object.

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoObjectContainer

getObject, setObject

Related Documentation

None

CiscoConnectionUniqueIDChangedEv

It's a new event to highlight that uniqueID of the connection has changed.

Interface History

Cisco Unified Communications Manager Release Number	Description
8.0(1)	New event

Declaration

public interface CiscoConnectionUniqueIDChangedEv extends ConnEv

Methods

Table 85: Methods in CiscoConnectionUniqueIDChangedEv

Interface	Method	Description
String	getOldUniqueID()	This method returns the old uniqueID of the connection which has just changed. The returned value is a Unique Identifier as 32-character hex string.
Terminal	getTerminal()	This method returns the Terminal for which this ConnEv is delivered.
String	getUniqueID()	This method returns the updated uniqueID of the connection. The returned value is a Unique Identifier as 32-character hex string.

Related Documentation

CiscoConsultCall

The CiscoConsultCall interface extends the CiscoCall interface to expose certain properties of calls that have been created as part of a consultative transfer or consultative conference.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.Call, javax.telephony.callcontrol.CallControlCall, CiscoCall, CiscoObjectContainer

Declaration

public interface CiscoConsultCall extends CiscoCall

Fields

None

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoCall

CALLSECURITY_AUTHENTICATED, CALLSECURITY_ENCRYPTED, CALLSECURITY_NOTAUTHENTICATED, CALLSECURITY_UNKNOWN, FEATUREPRIORITY_EMERGENCY, FEATUREPRIORITY_NORMAL, FEATUREPRIORITY_URGENT, PLAYTONE_BOTHLOCALANDREMOTE, PLAYTONE_LOCALONLY, PLAYTONE_NOLOCAL_OR_REMOTE, PLAYTONE_REMOTEONLY, SILENT_MONITOR

From Interface javax.telephony.Call

ACTIVE, IDLE, INVALID

Methods

Table 86: Methods in CiscoConsultCall

Interface	Method	Description
javax.telephony.TerminalConnection	getConsultingTerminalConnection()	Returns the consulting TerminalConnection that was used to create this CiscoConsultCall. If this Call was created as part of a consultative transfer or consultative conference, the getConsultingTerminalConnection method returns the TerminalConnection that was used to perform the consultation on the original call. This method lets you correlate a ConsultCall with its original call. The original call itself does not have any methods that you can use determine the ConsultCall, if any, to which it is related. Returns: Null if this Call does not result from a consultation, or the consulting TerminalConnection of the original Call if this call resulted from a consultation.

Interface	Method	Description
javax.telephony.Connection[]	consultWithoutMedia(javax.telephony.TerminalConnection tc, java.lang.String dialedDigits)	Provides applications ability to initiate a consultative call without setting up media for it. This interface may be invoked when application is creating a consult call and completing transfer before media establishes for consult call. Cisco Unified Communication Manager may some times run into erroneous race condition when consult call is answered, and application completes transfer in the middle of media setup for consult call. To avoid this problem, application that does not wait for media setup completion for consult call, may use this method to setup consult call. From CallEvent perspective, this method behaves similar to CallControlCall.consult(TerminalConnection tc, String dialedDigits). Creates a consultation between this Call and an active Call without establishing the media. This consult call may only be transferred, not conferenced. Cisco JTAPI does not support this method with CallControlCall.setConferenceEnable(). Cisco JTAPI only supports this method with CallControlCall.setTransferEnable(). **Throws** javax.telephony. InvalidStateException javax.telephony. InvalidArgumentException javax.telephony. MethodNotSupportedException javax.telephony. ResourceUnavailableException javax.telephony. PrivilegeViolationException javax.telephony. InvalidPartyException

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoCall

conference, connect, getCalledAddressPI, getCalledPartyInfo, getCallID, getCallingAddressPI, getCallSecurityStatus, getConferenceChain, getCurrentCalledAddress, getCurrentCalledAddressPI, getCurrentCalledDisplayNamePI, getCurrentCalledPartyDisplayName, getCurrentCalledPartyInfo, getCurrentCalledPartyUnicodeDisplayName, getCurrentCalledPartyUnicodeDisplayNamelocale, getCurrentCallingAddress, getCurrentCallingAddressPI, getCurrentCallingDisplayNamePI, getCurrentCallingPartyDisplayName, getCurrentCallingPartyInfo, getCurrentCallingPartyUnicodeDisplayName, getCurrentCallingPartyUnicodeDisplayNamelocale, getGlobalizedCallingParty, getLastRedirectedPartyInfo, getLastRedirectingAddressPI, getLastRedirectingPartyInfo, getModifiedCalledAddress, getModifiedCallingAddress, startMonitor, startMonitor, transfer

From Interface javax.telephony.callcontrol.CallControlCall

addParty, conference, consult, consult, drop, getCalledAddress, getCallingAddress, getCallingTerminal, getConferenceController, getConferenceEnable, getLastRedirectedAddress, getTransferController, getTransferEnable, offHook, setConferenceController, setConferenceEnable, setTransferController, setTransferEnable, transfer, transfer

From Interface javax.telephony.Call

addObserver, connect, getCallCapabilities, getCapabilities, getConnections, getObservers, getProvider, getState, removeObserver

From Interface com.cisco.jtapi.extensions.CiscoObjectContainer

getObject, setObject

Related Documentation

See CiscoCall for more information.

CiscoConsultCallActiveEv

The CiscoConsultCallActiveEv event interface extends the JTAPI CallActiveEv event. This event indicates that the state of the call object changed to Call.ACTIVE and that the call was initiated as a result of a consultative transfer or consultative conference operation (manual or programmatic). Applications can obtain the consulting TerminalConnection on the original (consulting) call by using the CiscoConsultCall.getConsultingTerminalConnection method.

The system reports this event to applications via the CallObserver interface.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.events.CallActiveEv, javax.telephony.events.CallEv, CiscoCallEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoConsultCallActiveEv extends CiscoCallEv, javax.telephony.events.CallActiveEv

Fields

None

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoCallEv

CAUSE_ACCESSINFORMATIONDISCARDED, CAUSE_BARGE, CAUSE_BCBPRESENTLYAVAIL, CAUSE_BCNAUTHORIZED, CAUSE_BEARERCAPNIMPL, CAUSE_CALLBEINGDELIVERED, CAUSE_CALLIDINUSE, CAUSE_CALLMANAGER_FAILURE, CAUSE_CALLREJECTED, CAUSE_CALLSPLIT, CAUSE_CHANTYPENIMPL, CAUSE_CHANUNACCEPTABLE, CAUSE_CTICCMSIP400BADREQUEST, CAUSE_CTICCMSIP401UNAUTHORIZED, CAUSE_CTICCMSIP402PAYMENTREQUIRED, CAUSE_CTICCMSIP403FORBIDDEN, CAUSE_CTICCMSIP404NOTFOUND, CAUSE_CTICCMSIP405METHODNOTALLOWED, CAUSE_CTICCMSIP406NOTACCEPTABLE, CAUSE_CTICCMSIP407PROXYAUTHENTICATIONREQUIRED, CAUSE_CTICCMSIP408REQUESTTIMEOUT, CAUSE_CTICCMSIP410GONE, CAUSE_CTICCMSIP411LENGTHREQUIRED, CAUSE_CTICCMSIP413REQUESTENTITYTOOLONG, CAUSE_CTICCMSIP414REQUESTURITOOLONG, CAUSE_CTICCMSIP415UNSUPPORTEDMEDIATYPE, CAUSE_CTICCMSIP416UNSUPPORTEDURISCHEME, CAUSE_CTICCMSIP420BADEXTENSION, CAUSE_CTICCMSIP421EXTENSTIONREQUIRED, CAUSE_CTICCMSIP423INTERVALTOOBRIEF, CAUSE_CTICCMSIP480TEMPORARILYUNAVAILABLE, CAUSE_CTICCMSIP481CALLLEGDOESNOTEXIST, CAUSE_CTICCMSIP482LOOPDETECTED, CAUSE_CTICCMSIP483TOOMANYHOOPS, CAUSE_CTICCMSIP484ADDRESSINCOMPLETE, CAUSE_CTICCMSIP485AMBIGUOUS, CAUSE_CTICCMSIP486BUSYHERE, CAUSE_CTICCMSIP487REQUESTTERMINATED, CAUSE_CTICCMSIP488NOTACCEPTABLEHERE, CAUSE_CTICCMSIP491REQUESTPENDING, CAUSE_CTICCMSIP493UNDECIPHERABLE, CAUSE_CTICCMSIP500SERVERINTERNALERROR, CAUSE_CTICCMSIP501NOTIMPLEMENTED, CAUSE_CTICCMSIP502BADGATEWAY, CAUSE_CTICCMSIP503SERVICEUNAVAILABLE, CAUSE_CTICCMSIP504SERVERTIMEOUT, CAUSE_CTICCMSIP505SIPVERSIONNOTSUPPORTED, CAUSE_CTICCMSIP513MESSAGETOOLARGE, CAUSE_CTICCMSIP600BUSYEVERYWHERE, CAUSE_CTICCMSIP603DECLINE, CAUSE_CTICCMSIP604DOESNOTEXISTANYWHERE, CAUSE_CTICCMSIP606NOTACCEPTABLE, CAUSE_CTICONFERENCEFULL, CAUSE_CTIDEVICENOTPREEMPTABLE, CAUSE_CTIDROPCONFEREE, CAUSE_CTIMANAGER_FAILURE, CAUSE_CTIPRECEDENCECALLBLOCKED, CAUSE_CTIPRECEDENCELEVELEXCEEDED, CAUSE_CTIPRECEDENCEOUTOFBANDWIDTH, CAUSE_CTIPREEMPTFORREUSE, CAUSE_CTIPREEMPTNOREUSE, CAUSE_DESTINATIONOUTOFORDER, CAUSE_DESTNUMMISSANDDCNOTSUB, CAUSE_DPARK, CAUSE_DPARK_REMINDER, CAUSE_DPARK_UNPARK, CAUSE_EXCHANGEROUTINGERROR, CAUSE_FAC_CMC, CAUSE_FACILITYREJECTED, CAUSE_IDENTIFIEDCHANDOESNOTEXIST, CAUSE_IENIMPL, CAUSE_INBOUNDBLINDTRANSFER, CAUSE_INBOUNDCONFERENCE, CAUSE_INBOUNDTRANSFER, CAUSE_INCOMINGCALLBARRED, CAUSE_INCOMPATABLEDDESTINATION, CAUSE_INTERWORKINGUNSPECIFIED, CAUSE_INVALIDCALLREFVALUE, CAUSE_INVALIDIECONTENTS, CAUSE_INVALIDMESSAGEUNSPECIFIED, CAUSE_INVALIDNUMBERFORMAT, CAUSE_INVALIDTRANSITNETSEL, CAUSE_MANDATORYIEMISSING, CAUSE_MSGNCOMPATABLEWCS, CAUSE_MSGTYPENCOMPATWCS, CAUSE_MSGTYPENIMPL, CAUSE_NETOUTOFORDER, CAUSE_NOANSWERFROMUSER, CAUSE_NOCALLSUSPENDED, CAUSE_NOCIRCAVAIL, CAUSE_NOERROR, CAUSE_NONSELECTEDUSERCLEARING, CAUSE_NORMALCALLCLEARING, CAUSE_NORMALUNSPECIFIED, CAUSE_NOROUTETODDESTINATION, CAUSE_NOROUTETOTRANSITNET, CAUSE_NOUSERRESPONDING, CAUSE_NUMBERCHANGED, CAUSE_ONLYRDIVEARERCAPAVAIL, CAUSE_OUTBOUNDCONFERENCE,

CAUSE_OUTBOUNDTRANSFER, CAUSE_OUTOFBANDWIDTH, CAUSE_PROTOCOLERRORUNSPECIFIED, CAUSE_QSIG_PR, CAUSE_QUALOFSERVNAVAIL, CAUSE_QUIET_CLEAR, CAUSE_RECOVERYONTIMEREXPIRY, CAUSE_REDIRECTED, CAUSE_REQCALLIDHASBEENCLEARED, CAUSE_REQCIRCNAVIL, CAUSE_REQFACILITYNIMPL, CAUSE_REQFACILITYNOTSUBSCRIBED, CAUSE_RESOURCESNAVAIL, CAUSE_RESPONSETOSTATUSENQUIRY, CAUSE_SERVNOTAVAILUNSPECIFIED, CAUSE_SERVOPERATIONVIOLATED, CAUSE_SERVOROPTNAVAILORIMPL, CAUSE_SUBSCRIBERABSENT, CAUSE_SUSPCALLBUTNOTTHISONE, CAUSE_SWITCHINGEQUIPMENTCONGESTION, CAUSE_TEMPORARYFAILURE, CAUSE_UNALLOCATEDNUMBER, CAUSE_USERBUSY

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

javax.telephony.TerminalConnectiongetHeldTerminalConnection() Deprecated. Replaced by CiscoConsultCall.getConsultingTerminalConnection()

Table 87: Methods in CiscoConsultCallActiveEv

Interface	Method	Description
javax.telephony.TerminalConnection	getHeldTerminalConnection()	Deprecated method. Replaced by CiscoConsultCall. GetConsultingTerminalConnection(). Returns the consulting TerminalConnection that was used to create this CiscoConsultCall. You can use this method to correlate a consultation call with its original call. The original call does not have any methods that you can use to determine the consultation call, if any, to which it is related. Returns: The consulting TerminalConnection of the call that created the call that is referenced by this event

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoCallEv

getCiscoCause, getCiscoFeatureReason

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Call, CallObserver, CallActiveEv and Constant Field Values for more information.

CiscoEv

The CiscoEv interface extends this code JTAPI javax.telephony.events.Ev interface and serves as the base interface for all Cisco-extended JTAPI events. Every event in this package extends this interface, directly or indirectly.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.events.Ev

Subinterfaces

CiscoAddrActivatedEv, CiscoAddrActivatedOnTerminalEv, CiscoAddrAddedToTerminalEv, CiscoAddrAutoAcceptStatusChangedEv, CiscoAddrCreatedEv, CiscoAddrEv, CiscoAddrInServiceEv, CiscoAddrIntercomInfoChangedEv, CiscoAddrIntercomInfoRestorationFailedEv, CiscoAddrOutOfServiceEv, CiscoAddrRecordingConfigChangedEv, CiscoAddrRemovedEv, CiscoAddrRemovedFromTerminalEv, CiscoAddrRestrictedEv, CiscoAddrRestrictedOnTerminalEv, CiscoCallChangedEv, CiscoCallEv, CiscoCallSecurityStatusChangedEv, CiscoConferenceChainAddedEv, CiscoConferenceChainRemovedEv, CiscoConferenceEndEv, CiscoConferenceStartEv, CiscoConsultCallActiveEv, CiscoMediaOpenLogicalChannelEv, CiscoOutOfServiceEv, CiscoProvCallParkEv, CiscoProvEv, CiscoProvFeatureEv, CiscoProvTerminalCapabilityChangedEv, CiscoRestrictedEv, CiscoRTPInputKeyEv, CiscoRTPInputStartedEv, CiscoRTPInputStoppedEv, CiscoRTPOutputKeyEv, CiscoRTPOutputStartedEv, CiscoRTPOutputStoppedEv, CiscoTermActivatedEv, CiscoTermButtonPressedEv, CiscoTermCreatedEv, CiscoTermDataEv, CiscoTermDeviceStateActiveEv, CiscoTermDeviceStateAlertingEv, CiscoTermDeviceStateHeldEv, CiscoTermDeviceStateIdleEv, CiscoTermDeviceStateWhisperEv, CiscoTermDNDOptionChangedEv, CiscoTermDNDStatusChangedEv, CiscoTermEv, CiscoTermInServiceEv, CiscoTermOutOfServiceEv, CiscoTermRegistrationFailedEv, CiscoTermRemovedEv, CiscoTermRestrictedEv, CiscoTermSnapshotCompletedEv, CiscoTermSnapshotEv, CiscoToneChangedEv, CiscoTransferEndEv, CiscoTransferStartEv

Declaration

public interface CiscoEv extends javax.telephony.events.Ev

Fields

None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Ev for more information.

CiscoFeatureReason

The CiscoFeatureReason interface specifies the feature reason that is associated with each delivered event.

Interface History

Cisco Unified Communications Manager Release	Description
7.1(1 and 2)	Added new reason, FORWARD_NO_RETRIEVE, for the Park Monitoring and Assisted DPark feature.
8.0(1)	A new reason code, REASON_SAF_CCD_PSTN_FAILOVER, has been added to convey the proper reason for a PSTN failover to the application.
8.5(1)	A new feature reason, REASON_MEDIA_STREAMING, is added.
10.0.1	A new feature reason, REASON_PLAY_ANNOUNCEMENT, is added.

Declaration

public interface CiscoFeatureReason

Fields

Table 88: Fields in CiscoFeatureReason

Interface	Field	Description
static int	REASON_BARGE	Indicates that the reason for the event is BARGE feature.
static int	REASON_BLINDTRANSFER	Indicates that reason is single step transfer
static int	REASON_CALLPICKUP	Indicates that the reason for the events is PICKUP
static int	REASON_CCM_REDIRECTION	Indicates that the reason for the events is SIP 3xx feature.
static int	REASON_CLICK_TO_CONFERENCE	Indicates that connections have been added or removed by using the Click to Conference feature
static int	REASON_CONFERENCE	Indicates that the reason for the event is CONFERENCE
static int	REASON_DPARK_CALLPARK	Indicates that the reason for events is DPARK feature
public static final int	REASON_DEQUEUING	Indicates that the event is gererated because the call has got de-queued under the Native Queuing Feature.
public static final int	REASON_DEQUEUING_TIMER_EXPIRED	Indicates that the event is generated because the call is de-queued under the Native Queuing Feature as the maximum queue timer expired.
public static final int	REASON_DEQUEUING_AGENTS_BUSY	Indicates that the event has been gererated because the call has got de-queued under the Native Queuing Feature as the agents were busy and the queue was full.
public static final int	REASON_DEQUEUING_AGENTS_UNAVAILABLE	Indicates that the event is gererated because the call has got de-queued under the Native Queuing Feature as the agents were either not logged-in or were unregistered.
static int	REASON_DPARK_REVERSION	Indicates that the reason for events in DPARK Reversion
static int	REASON_DPARK_UNPARK	Indicates that the reason for events in DPARK UNPARK
static int	REASON_FAC_CMC	Indicates that the reason for the events is FAC, CMC feature
static int	REASON_FORWARDALL	Indicates that reason for the event is FORWARD
static int	REASON_FORWARDBUSY	Indicates that the reasons for the event is forward busy
static int	REASON_FORWARDNOANSWER	Indicates that the reasons for the event is forward no answer

Interface	Field	Description
static int	REASON_FORWARD_NO_RETRIEVE	Indications that the reason for the event is forward no retrieve
static int	REASON_IMMDIVERT	Indicates that the reason for the events is imm divert
static int	REASON_MEDIA_STREAMING	Indicates that the event received is related to an Agent Greeting call
static int	REASON_MOBILITY	Indicates that the reason for events caused by Mobility Manager feature
static int	REASON_MOBILITY_CELLPICKUP	Indicates that the reason for events caused by Mobility Manager feature
static int	REASON_MOBILITY_FOLLOWME	Indicates that the reason for events caused by Mobility Manager feature
static int	REASON_MOBILITY_HANDIN	Indicates that the reason for events caused by Mobility Manager feature
static int	REASON_MOBILITY_HANDOUT	Indicates that the reason for events caused by Mobility Manager feature
static int	REASON_MOBILITY_IVR	Indicates that the reason for events caused by Mobility Manager feature
static int	REASON_NORMAL	Indicates that the reason for the event is NORMAL
static int	REASON_PARK	Indicates that the reason for the event is PARK feature
static int	REASON_PARKREMAINDER	Indicates that the reasons for the event is park remainder
public static final int	REASON_PLAY_ANNOUNCEMENT	This interface indicates that the event was generated because of a play announcement.
static int	REASON_QSIG_PR	Indicates that the reason for the event is QSIG path replacement
public static final int	REASON_QUEUING	Indicates that the event is generated due to the Native Queuing feature.
static int	REASON_REDIRECT	Indicates that the reason for event is REDIRECT
static int	REASON_REFER	Returned for events sent for REFER done at Cisco Unified Communications Manager
static int	REASON_REPLACE	REASON_REPLACE : This reason will be returned for events send for REPLACE feature done at Cisco Unified Communications Manager
static int	REASON_SILENTMONITORING	Indicates that the reason for events in SILENT MONITORING

Interface	Field	Description
static int	REASON_TRANSFER	Indicates that the reason for the event is TRANSFER
static int	REASON_UNPARK	Indicates that the reason for the event is unpark
static final int	REASON_SAF_CCD_PSTN_FAILOVER	Indicates the reason for PSTN failover to the application

Related Documentation

See Constant Field Values for more information.

CiscoHuntConnection

A CiscoHuntConnection in a call indicates that the call is routed through a hunt pilot.

Interface History

Cisco Unified Communications Manager Release Number	Description
8.0(1)	New interface

Declaration

public interface CiscoHuntConnection extends CiscoConnection.

Methods

Table 89: Methods in CiscoHuntConnection

Interface	Method	Description
Connection[]	getAgentConnections()	This method returns an array of connections to the hunt group member or null.

Related Documentation

CiscoIntercomAddress

The CiscoIntercomAddress interface extends the CiscoAddress interface with additional Cisco Unified Communications Manager-specific capabilities for intercom addresses. This interface lets applications initiate intercom calls and take advantage of other intercom-specific features.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.Address, CiscoAddress, CiscoObjectContainer

Declaration

public interface CiscoIntercomAddress extends CiscoAddress

Fields

None

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoAddress

APPLICATION_CONTROLLED_RECORDING, AUTO_RECORDING, AUTOACCEPT_OFF, AUTOACCEPT_ON, AUTOANSWER_OFF, AUTOANSWER_UNKNOWN, AUTOANSWER_WITHHEADSET, AUTOANSWER_WITHSPEAKERSET, EXTERNAL, EXTERNAL_UNKNOWN, IN_SERVICE, INTERNAL, MONITORING_TARGET, NO_RECORDING, OUT_OF_SERVICE, RINGER_DEFAULT, RINGER_DISABLE, RINGER_ENABLE, UNKNOWN

Methods

Table 90: Methods in CiscoIntercomAddress

Interface	Method	Description
void	setIntercomTarget(java. lang. String targetDNjava. lang. String targetAsciiLabel, java. lang. String targetUnicodeLabel)	Sets the intercom target DN, intercom target label, and intercom target Unicode label that appears next to the intercom line on the phone. The phone displays the Unicode label if the phone has that capability; otherwise, the phone displays the ASCII target label. **Throws** javax. telephony. InvalidPartyException means that the target DN is invalid. javax. telephony. InvalidStateException means that the address, terminal, or provider are not in service. **Parameters** • targetDN—Destination DN for the intercom call • targetAsciiLabel—ASCII display label shown next to the intercom line on the phone target • UnicodeLabel—Unicode display label shown on the phone
Boolean	isIntercomTargetSet()	Returns true if an application has overridden the current value, or false if the current value matches the default value configured in the database.
void	resetIntercomTarget()	Resets the intercom target DN, intercom target label, and intercom target Unicode label to their default values. **Throws** javax. telephony. InvalidPartyException javax. telephony. InvalidStateException
java. lang. String	getIntercomTargetNumber()	Returns the current intercom target DN that the application set. If the application has not set the intercom target DN, this interface returns the default intercom target DN that is configured in Cisco Unified Communications Manager Administration. Returns: The intercom target DN number, as a string.
java. lang. String	getIntercomTargetAsciiLabel()	Returns the current intercom target label that the application set. If the application has not set the intercom target label, this interface returns the default intercom target label that is configured in Cisco Unified Communications Manager Administration. Returns: The intercom target label string.

Interface	Method	Description
java. lang. String	getIntercomTargetUnicodeLabel()	Returns the current intercom target Unicode label that the application set. If the application has not set the Unicode label, this interface returns the default intercom target Unicode label that is configured in Cisco Unified Communications Manager Administration. Returns: The intercom Unicode target label string.
java. lang. String	getDefaultIntercomTargetNumber()	Returns the default intercom target DN that is configured through Cisco Unified Communications Manager Administration. Returns: The default intercom target DN number, as a string.
java. lang. String	getDefaultIntercomTargetAsciiLabel()	Returns the default intercom target label that is configured through Cisco Unified Communications Manager Administration. Returns: The default intercom target label string.
java. lang. String	getDefaultIntercomTargetUnicodeLabel()	Returns the default intercom target label that is configured through Cisco Unified Communications Manager Administration. Returns: The default unicode intercom target label string.
javax. telephony. Connection[]	connectIntercom(javax. telephony. Terminal terminal, java. lang. String targetNumber)	Places an intercom call from an originating intercom address to a destination intercom address. Returns:A connection list for the calling and called intercom addresses. **Throws** javax. telephony. InvalidPartyException—The target DN is not a valid number. javax. telephony. InvalidArgumentException—The address is not a CiscoIntercomAddress or the terminal is not a Terminal. javax. telephony. InvalidStateException—The address, terminal, or provider is not in service. javax. telephony. ResourceUnavailableException—A resource is not available to complete the operation. javax. telephony. PrivilegeViolationException—The application does not have sufficient privileges to execute this operation.

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoAddress

clearCallConnections, getAddressCallInfo, getAutoAcceptStatus, getAutoAnswerStatus, getInServiceAddrTerminals, getPartition, getRecordingConfig, getRegistrationState,

getRestrictedAddrTerminals, getState, getType, isRestricted, setAutoAcceptStatus, setMessageWaiting, setRingerStatus

From Interface javax.telephony.Address

addCallObserver, addObserver, getAddressCapabilities, getCallObservers, getCapabilities, getConnections, getName, getObservers, getProvider, getTerminals, removeCallObserver, removeObserver

From Interface com.cisco.jtapi.extensions.CiscoObjectContainer

getObject, setObject

Related Documentation

See CiscoAddress for additional information.

CiscoIsacMediaCapability

The CiscoIsacMediaCapability object specifies the properties for a iSAC encoded RTP stream. Applications that support iSAC media termination use this object when registering a CiscoMediaTerminal.

The packet size and bit rate are variable.

Interface History

Cisco Unified Communications Manager Release Number	Description
8.0(1)	Interface added in this release for iSac codec which can be used by application to register a CiscoMediaTerminal or CiscoRouteTerminal if they want to use this new MediaCapability.

Superinterfaces

None

Declaration

public class CiscoIsacMediaCapabilityextends CiscoMediaCapability

Constuctors

Table 91: Constructor in CiscoIsacMediaCapability

Interface	Constructor	Description
public	CiscoIsacMediaCapability()	Constructs a CiscoIsacMediaCapability

Fields

None

Inherited Fields

From Class com.cisco.jtapi.extensions.CiscoMediaCapability

G711_64K_30_MILLISECONDS, G723_6K_30_MILLISECONDS, G729_30_MILLISECONDS, GSM_80_MILLISECONDS, ISAC, WIDEBAND_256K_10_MILLISECONDS

Methods

None

Inherited Methods

Inherited From Class com.cisco.jtapi.extensions.CiscoMediaCapability

getMaxFramesPerPacket, getPayloadType, isSupported, toString

Inherited From Class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

CiscoJtapiException

The CiscoJtapiException interface defines error codes that CTI requests may return. Cisco JTAPI extends all of the JTAPI exceptions to implement this interface. You can get the error codes by casting the exception to CiscoJtapiException and calling the method getErrorCode().

For example, if "e" is any exception caught by an application, the following code checks whether the exception is an instance of CiscoJtapiException.

```
try {
   // some code here
 }
catch ( Exception e ) {
   if( e instanceof CiscoJtapiException){
     CiscoJtapiException ce  =
com.cisco.cti.client.CTIFAILURE.(CiscoJtapiException) e
     int errorCode  =  com.cisco.cti.client.CTIFAILURE.ce.getErrorCode()
                     //returns the ErrorCode.
   }
 }
```

Interface History

Cisco Unified Communications Manager Release Number	Description
7.0	Added this interface.
7.1(1 and 2)	Added new error codes for the Logical Partition feature called: CiscoJtapiException.CTIERR_REDIRECT_CALL_PARTITIONING_POLICY and CiscoJtapiException.CTIERR_FEATURE_NOT_AVAILABLE.
8.0(1)	A new error code is added which will be exposed when the monitoring/recording request is rejected when the supervisor/recorder does not meet the security capabilities of the agent. New error code, OPERATION_NOT_AVAILABLE_IN_CURRENT_STATE, has also been added.
8.5(1)	The following new error codes are added:CTIERR_MEDIA_CONNECTION_FAILED, CTIERR_REQUEST_ALREADY_PENDING, CTIERR_START_STREAM_MEDIA_FAILED, CTIERR_STOP_STREAM_MEDIA_FAILED, CTIERR_NO_STREAMING_MEDIA_SESSION, CTIERR_EXISTING_STREAMING_MEDIA_SESSIONCTIERR_ MEDIA_ALREADY_TERMINATED_STATIC_GETPORT_SUPPORT, CTIERR_MEDIA_ALREADY_TERMINATED_DYNAMIC_GETPORT_SUPPORT, CTIERR_SSO_DISABLED, CTIERR_SSO_AUTH_SERVER_DOWN.
9.0(1)	The following new error codes are added:CTIERR_INVALID_REMOTE_DESTINATION_NUMBER CTIERR_DUPLICATE_REMOTE_DESTINATION_NUMBER CTIERR_REMOTEDESTINATION_LIMIT_EXCEEDED CTIERR_REMOTE_DEVICE_REQUEST_FAILED_ACTIVE_RD_NOT_SET CTIERR_ENDUSER_NOT_ASSOCIATED_WITH_DEVICE CTIERR_DEVICE_ALREADY_REGISTERED_NONEXTEND CTIERR_MEDIA_ALREADY_TERMINATED_EXTEND CTIERR_INVALID_REMOTE_DESTINATION_NAME CTIERR_RECORDING_INVOCATION_TYPE_NOT_MATCHING

Cisco Unified Communications Manager Release Number	Description
10.0(1)	The following new error codes are added: CTIERR_AUTHENTICATION_TYPE_ON_UNSUPPORTED_PORT CTIERR_NO_PERSISTENT_CALL_EXISTS CTIERR_ANNOUNCEMENT_ALREADY_IN_PROGRESS CTIERR_ERROR_PLAYING_ANNOUNCEMENT CTIERR_PLAY_ANNOUNCEMENT_FAILED CTIERR_EXTEND_AND_CONNECT_DESTINATION_NOT_REACHABLE CTIERR_CREATE_PERSISTENT_CALL_FAILED CTIERR_PERSISTENT_CALL_EXISTS CTIERR_OPERATION_NOT_ALLOWED_ON_PERSISTENT_CALL CTIERR_DISCONNECT_PERSISTENT_CALL_FAILED_CALL_ACTIVE CTIERR_PERSISTENT_CALL_BEING_SETUP CTIERR_INVALID_SSO_TOKEN_SIZE

Declaration

public interface CiscoJtapiException

Fields

Table 92: Fields in CiscoJtapiException

Interface	Field	Description
static int	ASSOCIATED_LINE_NOT_OPEN	This error indicates that the request is issued on a line, which is not open
static int	CALL_ALREADY_EXISTS	This error indicates that another call already exists on the line
static int	CALL_DROPPED	The call dropped after the feature request (hold, unhold, transfer, or conference) but before the request was completed.
static int	CALLHANDLE_NOTINCOMINGCALL	This error indicates that an attempt is made to answer a call that either does not exist or is not in the correct state
static int	CALLHANDLE_UNKNOWN_TO_ LINECONTROL	This error indicates that attempt to redirect call that was unknown to line control

Interface	Field	Description
static int	CANNOT_OPEN_DEVICE	This error indicates that device open failed because the associated device is unregistering
static int	CANNOT_TERMINATE_MEDIA_ON_PHONE	This error indicates that media cannot be terminated by an application when the device is a physical phone (the phone always terminates the media)
static int	CFWDALL_ALREADY_SET	This error indicates that attempt to set CFWALL while it is already set
static int	CFWDALL_DESTN_INVALID	This error indicates that attempt to CFWALL to an invalid destination
static int	CLUSTER_LINK_FAILURE	This error indicates that link to one of the cisco unified communications managers failed in the cluster (network error)
static int	COMMAND_NOT_IMPLEMENTED_ON_DEVICE	This error indicates that device does not support the command.
static int	CONFERENCE_ALREADY_PRESENT	This error indicates that attempt to conference a party that is already in conference
static int	CONFERENCE_FAILED	This error indicates that conference completion was not successful.
static int	CONFERENCE_FULL	This error indicates that all conference bridges are busy.
static int	CONFERENCE_INACTIVE	This error indicates that attempt to complete conference while consult conference is not active
static int	CONFERENCE_INVALID_PARTICIPANT	This error indicates that an attempt to conference to self or an invalid participant
static int	CTIERR_ACCESS_TO_DEVICE_DENIED	This error indicates that the access to device is denied.
public static final int	CTIERR_ANNOUNCEMENT_ALREADY_IN_PROGRESS	This error indicates that the announcement is already in progress.
static int	CTIERR_APP_SOFTKEYS_ALREADY_CONTROLLED	This error indicates that the application softkeys are already controlled by another application
static int	CTIERR_APPLICATION_DATA_SIZE_EXCEEDED	This error indicates that application data size has exceeded limit
static int	CTIERR_AUTHENTICATION_TYPE_ON_UNSUPPORTED_PORT	This error indicates that the port does not support the authentication type that is used to create the provider.
static int	CTIERR_BIB_NOT_CONFIGURED	This error indicates built in bridge is not configured

Interface	Field	Description
static int	CTIERR_BIB_RESOURCE_NOT_ AVAILABLE	This error indicates that built in bridge resource not available
static int	CTIERR_CALL_MANAGER_NOT_ AVAILABLE	This error indicates that Communications Manager is not available currently
static int	CTIERR_CALL_NOT_EXISTED	This error indicates that call does not exist
static int	CTIERR_CALL_PARK_NO_DN	This error indicates no call park DN
static int	CTIERR_CALL_REQUEST_ALREADY_ OUTSTANDING	This error indicates call request already outstanding
static int	CTIERR_CALL_UNPARK_FAILED	This error indicates that call unpark did not succeed
static int	CTIERR_CAPABILITIES_DO_NOT_MATCH	This error indicates that capabilities do not match
static int	CTIERR_CLOSE_DELAY_NOT_ SUPPORTED_WITH_REG_TYPE	This error indicates that the close delay is not supported with this registration type
static int	CTIERR_CONFERENCE_ALREADY_ EXISTED	This error indicates that conference already exists
static int	CTIERR_CONFERENCE_NOT_EXISTED	This error indicates that conference does not exist
static int	CTIERR_CONNECTION_ON_INVALID_PORT	This error indicates application is trying to connect to invalid port
static int	CTIERR_CONSULT_CALL_FAILURE	This error indicates consult call failure
static int	CTIERR_CONSULTCALL_ALREADY_ OUTSTANDING	This error indicates that consult call already outstanding
static int	CTIERR_CRYPTO_CAPABILITY_MISMATCH	This error indicates that device registration failed as device crypto algorithms does not match with current device registration
static int	CTIERR_CTIHANDLER_PROCESS_ CREATION_FAILED	This error indicates that CTIHandler process creation failed
static int	CTIERR_DB_INITIALIZATION_ERROR	This error indicates DB initialization error
static int	CTIERR_DEVICE_ALREADY_OPENED	This error indicates that device is already opened
static int	CTIERR_DEVICE_NOT_OPENED_YET	This error indicates that device is not yet opened
static int	CTIERR_DEVICE_OWNER_ALIVE_ TIMER_STARTED	This error indicates that there is a device registration failure
static int	CTIERR_DEVICE_REGISTRATION_FAILED_ NOT_SUPPORTED_ MEDIATYPE	This error indicates an invalid media type, CTIPort need to be registered with Dynamic media port registation if it has an intercom line
static int	CTIERR_DEVICE_RESTRICTED	This error indicates that the device is restricted

Interface	Field	Description
static int	CTIERR_DEVICE_SHUTTING_DOWN	This error indicates that device is shutting down
static int	CTIERR_DIRECTORY_LOGIN_TIMEOUT	This error indicates that there is a directory login time out
static int	CTIERR_DUPLICATE_CALL_REFERENCE	This error indicates duplicate call reference
static int	CTIERR_DYNREG_IPADDRMODE_ MISMATCH	This indicates registration failure when Cisco Media/Route Terminal is already registered with different Addressing mode.
public static final int	CTIERR_ERROR_PLAYING_	This error indicates that there was an error playing the announcement.
static int	CTIERR_EXISTING_STREAMING_ MEDIA_SESSION	This error occurs if an application attempts to invoke an Agent Greeting API while another request is made and accepted. JTAPI throws InvalidStateException with a description as “There is an existing streaming media session”.
public static final int	CTIERR_EXTEND_AND_CONNECT_ DESTINATION_NOT_REACHABLE	This error occurs if the extend and connect destination is not reachable.
static int	CTIERR_FAC_CMC_REASON_CMC_ INVALID	Client Matter Code (CMC) entered is invalid
static int	CTIERR_FAC_CMC_REASON_CMC_ NEEDED	CMC is required to offer the call
static int	CTIERR_FAC_CMC_REASON_FAC_ CMC_NEEDED	Forced Authorization Code (FAC) and CMC are required to offer call
static int	CTIERR_FAC_CMC_REASON_FAC_INVALID	FAC entered is invalid
static int	CTIERR_FAC_CMC_REASON_FAC_ NEEDED	FAC is required to offer the call
static int	CTIERR_FEATURE_ALREADY_REGISTERED	This error indicates feature already registered
static int	CTIERR_FEATURE_DATA_REJECT	This error indicates feature data reject
static int	CTIERR_FEATURE_SELECT_FAILED	This error indicates that feature select failed
static int	CTIERR_ILLEGAL_DEVICE_TYPE	This error indicates that the device type is illegal
static int	CTIERR_INCOMPATIBLE_AUTOINSTALL_ PROTOCOL_VERSION	This error indicates that auto install protocol version is incompatible
static int	CTIERR_INCORRECT_MEDIA_CAPABILITY	Device registration failed due to incorrect media capability.
static int	CTIERR_INFORMATION_NOT_AVAILABLE	This error indicates that information is not available

Interface	Field	Description
static int	CTIERR_INTERCOM_SPEEDDIAL_ ALREADY_CONFIGURED	This error indicates that intercom target value is already configured, application is trying to make call with Intercom target DN
static int	CTIERR_INTERCOM_SPEEDDIAL_ ALREADY_SET	This error indicates that intercom request failed as intercom target value is already set, application is trying to set again
static int	CTIERR_INTERCOM_SPEEDDIAL_ DESTN_INVALID	This error indicates that intercomm request failed as intercom target value in not in the intercom group
static int	CTIERR_INTERCOM_TALKBACK_ ALREADY_PENDING	This error indicates that intercom talk back request is already pending
static int	CTIERR_INTERCOM_TALKBACK_FAILURE	This error indicates that talkback request failed for some reason.
static int	CTIERR_INTERNAL_FAILURE	This error indicates there is a CTI internal failure
static int	CTIERR_INVALID_CALLID	This error indicates the call ID is invalid
static int	CTIERR_INVALID_DEVICE_NAME	This error indicates that the device name is not valid
static int	CTIERR_INVALID_DTMFDIGITS	Play DTMF request failed because it is an invalid DTMF digit.
static int	CTIERR_INVALID_FILTER_SIZE	This error indicates that filter size is invalid
static int	CTIERR_INVALID_MEDIA_DEVICE	This error indicates that the media device is not valid
static int	CTIERR_INVALID_MEDIA_PARAMETER	This error indicates media parameter is inavlid
static int	CTIERR_INVALID_MEDIA_PROCESS	This error indicates that there is an invalid media process
static int	CTIERR_INVALID_MEDIA_RESOURCE_ID	This error indicates media resource ID is not valid
static int	CTIERR_INVALID_MESSAGE_HEADER_INFO	This error indicates that the header info is not valid
static int	CTIERR_INVALID_MESSAGE_LENGTH	This error indicates that message length is invalid
static int	CTIERR_INVALID_MONITOR_DESTN	This error indicates monitoring request failed due to invalid monitoring destination
static int	CTIERR_INVALID_MONITOR_DN_TYPE	This error indicates an invalid monitor DN type
static int	CTIERR_INVALID_MONITORMODE	This error indicates monitor request failed due to an invalid monitor mode
static int	CTIERR_INVALID_PARAMETER	This error indicates that the parameter is not valid
static int	CTIERR_INVALID_PARK_DN	This error indicates that the DN is an invalid park DN

Interface	Field	Description
static int	CTIERR_INVALID_PARK_REGISTRATION_ HANDLE	This error indicates that the handle is an invalid park registration handle
static int	CTIERR_INVALID_RESOURCE_TYPE	This error indicates an invalid resource type
static int	CTIERR_IPADDRMODE_MISMATCH	This indicates the registration failure due to IP Addressing Mode mismatch.
static int	CTIERR_LINE_OUT_OF_SERVICE	This error indicates that line is out of service.
static int	CTIERR_LINE_RESTRICTED	This error indicates that the line is restricted
static int	CTIERR_MAXCALL_LIMIT_REACHED	This error indicates that maximum call limit has reached
static int	CTIERR_MEDIA_ALREADY_TERMINATED_ DYNAMIC	This error indicates that device registration failed as device is registered with Dynamic media termination
Final static int	CTIERR_MEDIA_ALREADY_TERMINATED_ DYNAMIC_GETPORT_SUPPORT	This error indicates that the application tries to register a terminal, which is already registered with get port support, with a different registration type.
static int	CTIERR_MEDIA_ALREADY_TERMINATED_ NONE	This error indicates that device registration failed as device is already registered with media termination none
static int	CTIERR_MEDIA_ALREADY_TERMINATED_ STATIC	This error indicates that device registration failed as device is registered with Static media termination
Final static int	CTIERR_MEDIA_ALREADY_TERMINATED_ STATIC_GETPORT_SUPPORT	This error indicates that the application tries to register a terminal, which is already registered with get port support, with a different registration type.
static int	CTIERR_MEDIA_CAPABILITY_MISMATCH	This error indicates that device registration failed as media capability of device does not match with current device registration
static int	CTIERR_MEDIA_CONNECTION_FAILED	This error indicates that there is a general failure with the Agent Greeting feature. JTAPI throws InvalidStateException with a description as “The connection to the media has failed”.
static int	CTIERR_MEDIA_RESOURCE_NAME_ SIZE_EXCEEDED	This error indicates that media resource name size has exceeded limit
static int	CTIERR_MEDIAREGISTRATIONTYPE_ DO_NOT_MATCH	This error indicates that media registration types do not match
static int	CTIERR_MESSAGE_TOO_BIG	This error indicates that message is too big
static int	CTIERR_MORE_ACTIVE_CALLS_THAN_ RESERVED	This error indicates that there are more active calls than reserved

Interface	Field	Description
static int	CTIERR_NO_EXISTING_CALLS	This error indicates there are no existing calls
static int	CTIERR_NO_EXISTING_CONFERENCE	This error indicates that there is no existing conference
public static final int	CTIERR_NO_PERSISTENT_CALL_EXISTS	This error indicates that no persistent call exists.
static int	CTIERR_NO_RECORDING_SESSION	This error indicates recording request failed as there is no recording session
static int	CTIERR_NO_RESPONSE_FROM_MP	This error indicates no response from media resource
static int	CTIERR_NO_STREAMING_MEDIA_SESSION	This error occurs if an application attempts to invoke a stop request while there is no existing media stream to stop. JTAPI throws InvalidStateException with a description as “There is no streaming media session active”.
static int	CTIERR_NOT_PRESERVED_CALL	This error indicates that the call is not preserved
static int	CTIERR_OPERATION_FAILED_QUIETCLEAR	This error indicates that feature unavailable for this call due to temporary failure
static int	CTIERR_OPERATION_NOT_ALLOWED	This error indicates that this operation is not allowed
static int	CTIERR_OUT_OF_BANDWIDTH	This error indicates out of bandwidth error
static int	CTIERR_OWNER_NOT_ALIVE	This error indicates a failure during registering the device
static int	CTIERR_PENDING_ACCEPT_OR_ ANSWER_REQUEST	This error indicates that there is a pending accept or answer request
static int	CTIERR_PENDING_START _MONITORING_REQUEST	This error indicates there is a pending start monitoring request
static int	CTIERR_PENDING_START_RECORDING_ REQUEST	This error indicates there is a pending start recording request
static int	CTIERR_PENDING_STOP_RECORDING_ REQUEST	This error indicates there is a pending stop recording request
public static final int	CTIERR_PLAY_ANNOUNCEMENT_FAILED	This error indicates that the play announcement failed.
static int	CTIERR_PRIMARY_CALL_INVALID	This error indicates that primary call in monitoring request in invalid or gone idle
static int	CTIERR_PRIMARY_CALL_STATE_INVALID	This error indicates that primary call in monitoring request is in invalid state
static int	CTIERR_RECORDING_ALREADY_ INPROGRESS	This error indicates recording request failed that recording is already in progress

Interface	Field	Description
static int	CTIERR_RECORDING_CONFIG_NOT_ MATCHING	This error indicates recording configuration does not match
static int	CTIERR_RECORDING_SESSION_INACTIVE	This error indicates recording request failed because recording session is inactive
static int	CTIERR_REDIRECT_UNAUTHORIZED_ COMMAND_USAGE	This error indicates a redirect unauthorized command usage
static int	CTIERR_REGISTER_FEATURE_ ACTIVATION_FAILED	This error indicates that register feature activation failed
static int	CTIERR_REGISTER_FEATURE_APP_ ALREADY_REGISTERED	Register feature application was already registered
static int	CTIERR_REGISTER_FEATURE_PROVIDER_ NOT_REGISTERED	Register feature provider was not registered.
static int	CTIERR_REQUEST_ALREADY_PENDING	This error occurs if an application attempts to invoke an Agent Greeting API while another request is made. JTAPI throws InvalidStateException with a description as “The request was rejected because there is a similar request already pending”.
static int	CTIERR_RESOURCE_NOT_AVAILABLE	This error indicates that resource is not available to fulfil the request
public static final int	CTIERR_SECURITY_CAPABILITY_MISMATCH	This error code is exposed when the monitoring/recording request is rejected when the supervisor/recorder does not meet the security capabilities of the agent.
int	CTIERR_SSO_AUTH_SERVER_DOWN	This error code is returned if authorization server is down.
int	CTIERR_SSO_DISABLED	This error code is returned if the Single Sign-On feature is not enabled on Cisco Unified Communications Manager.
static int	CTIERR_START_MONITORING_FAILED	This error indicates that start monitoring request failed
static int	CTIERR_START_RECORDING_FAILED	This error indicates that start recording request failed
static int	CTIERR_START_STREAM_MEDIA_FAILED	This error occurs if there is a general failure with the Agent Greeting feature, that is not covered by any of the other error codes.JTAPI throws InvalidStateException with a description as “Start streaming media request failed”.

Interface	Field	Description
static int	CTIERR_STOP_STREAM_MEDIA_FAILED	This error occurs if there is a general failure with the Agent Greeting feature, that is not covered by any of the other error codes. JTAPI throws InvalidStateException with a description as “Stop streaming media request failed”.
static int	CTIERR_STATION_SHUT_DOWN	This error indicates that there is a station shutdown
static int	CTIERR_SYSTEM_ERROR	This error indicates CTI system error
static int	CTIERR_UDP_PASS_THROUGH_NOT_ SUPPORTED	This error indicates UDP data passthrough not supported
static int	CTIERR_UNKNOWN_EXCEPTION	This error indicates an unknown exception occured
static int	CTIERR_UNSUPPORTED_CALL_PARK_ TYPE	This error indicates that call park type is not supported
static int	CTIERR_UNSUPPORTED_CFWD_TYPE	This error indicates that the call forward type is unsupported
static int	CTIERR_USER_NOT_AUTH_FOR_SECURITY	This error indicates user is not authorized for secure connection
static int	CTIERR_REDIRECT_CALL_PARTITIONING_ POLICY	This error indicates redirect is not authorized
static int	CTIERR_FEATURE_NOT_AVAILABLE	This error indicates that the feature is unavailable
static int	DARES_INVALID_REQ_TYPE	This error indicates that there is an internal call processing error: DaRes invalid request type
static int	DATA_SIZE_LIMIT_EXCEEDED	This error indicates that XML data object size is bigger than allowed.
static int	DB_ERROR	This error indicates that the device query contained an illegal device type
static int	DB_ILLEGAL_DEVICE_TYPE	This error indicates illegal device type in DB
static int	DB_NO_MORE_DEVICES	This error is no longer used.
static int	DESTINATION_BUSY	This error indicates that destination is busy
static int	DESTINATION_UNKNOWN	This error indicates that destination is not found
static int	DEVICE_ALREADY_REGISTERED	This error indicates that device registration attempt failed, because the device is already registered
static int	DEVICE_NOT_OPEN	This error indicates that an attempt to open a line failed, as the device is not opened or the device is not registered.

Interface	Field	Description
static int	DEVICE_OUT_OF_SERVICE	This error indicates that device is out of service.
static int	DIGIT_GENERATION_ALREADY_IN_ PROGRESS	This error indicates that digit generation is already in progress.
static int	DIGIT_GENERATION_CALLSTATE_ CHANGED	This error indicates that call state is invalid to continue.
static int	DIGIT_GENERATION_WRONG_CALL_ HANDLE	This error indicates that call handle is invalid and call may be gone.
static int	DIGIT_GENERATION_WRONG_CALL_ STATE	This error indicates that call state is not valid to generate digits.
static int	DIRECTORY_LOGIN_FAILED	This error indicates that directory login failed: directory not initialized
static int	DIRECTORY_LOGIN_NOT_ALLOWED	This error indicates that directory login failed
static int	DIRECTORY_TEMPORARY_UNAVAILABLE	This error indicates that directory is temporarily unavailable.
static int	EXISTING_FIRSTPARTY	This error indicates that there is already a device controlling media.
static int	HOLDFAILED	This error indicates that the hold was rejected by line control or call control layers
static int	ILLEGAL_CALLINGPARTY	This error indicates that an attempt was made to originate call using a calling party that is not on the device
static int	ILLEGAL_CALLSTATE	This error indicates line is not in a legal state to invoke the request
static int	ILLEGAL_HANDLE	This error indicates the handle is not valid
static int	ILLEGAL_MESSAGE_FORMAT	This error indicates that there is a QBE protocol error
static int	INCOMPATIBLE_PROTOCOL_VERSION	This error indicates that JTAPI and CTI versions are not compatible : CTI Error Protocol version not supported
static int	INVALID_LINE_HANDLE	This error indicates that attempt to perform a line operation on an invalid line handle.
static int	INVALID_RING_OPTION	This error indicates that the ring option is invalid
static int	LINE_GREATER_THAN_MAX_LINE	This error indicates that line is greater than the maximum available lines on this device
static int	LINE_INFO_DOES_NOT_EXIST	This error indicates that line information does not exist in the database.

Interface	Field	Description
static int	LINE_NOT_PRIMARY	This error indicates that internal error returned from call control.
static int	LINECONTROL_FAILURE	This error indicates line control refuses to allow a new call to be initiated because of its current state.
static int	MAX_NUMBER_OF_CTI_CONNECTIONS_ REACHED	The maximum number of CTI connections was reached.
static int	MSGWAITING_DESTN_INVALID	This error indicates that attempt to set message waiting lamp for an invalid DN; Message Waiting Destination not found.
static int	NO_ACTIVE_DEVICE_FOR_THIRDPARTY	This error indicates there is no active device for thirdparty
static int	NO_CONFERENCE_BRIDGE	This error indicates that no conference bridge available
static int	NOT_INITIALIZED	This error indicates that attempt is made to open a provider before CTI initialization completes
static int	OPERATION_NOT_AVAILABLE_IN_ CURRENT_STATE	This error indicates that the operation is not available in Current state.
static int	PROTOCOL_TIMEOUT	Internal error returned from call control
static int	PROVIDER_ALREADY_OPEN	This error indicates that an attempt is made to reopen provider
static int	PROVIDER_CLOSED	Attempt to close provider while it is already closed
static int	PROVIDER_NOT_OPEN	This error indicates that device list incomplete or device list query timeout or query aborted
static int	REDIRECT_CALL_CALL_TABLE_FULL	This error indicates that internal error is returned from call control
static int	REDIRECT_CALL_DESTINATION_BUSY	This error indicates that the redirect destination is busy
static int	REDIRECT_CALL_DESTINATION_OUT_ OF_ORDER	This error indicates that redirect destination is out of order
static int	REDIRECT_CALL_DIGIT_ANALYSIS_ TIMEOUT	This error indicates a digit analyss time out, this is an internal error returned from call control
static int	REDIRECT_CALL_DOES_NOT_EXIST	This error indicates that an attempt is made to redirect a call that does not exist or is not longer active
static int	REDIRECT_CALL_INCOMPATIBLE_STATE	This error indicates that internal error is returned from call control

Interface	Field	Description
static int	REDIRECT_CALL_MEDIA_CONNECTION_ FAILED	This error indicates media connection failure, this is an internal error returned from call control
static int	REDIRECT_CALL_NORMAL_CLEARING	This error indicates that redirect failed because of normal call clearing
static int	REDIRECT_CALL_ORIGINATOR_ ABANDONED	This error indicates that far end hung up on the call being redirected
static int	REDIRECT_CALL_PARTY_TABLE_ FULL	This error indicates that internal error is returned from call control
static int	REDIRECT_CALL_PENDING_REDIRECT_ TRANSACTION	This error indicates that internal error is returned from call control
static int	REDIRECT_CALL_PROTOCOL_ERROR	This error indicates a protocol error, this is an internal error returned from call control
static int	REDIRECT_CALL_UNKNOWN_DESTINATION	This error indicates that an attempt is made to redirect to an unknown destination
static int	REDIRECT_CALL_UNKNOWN_ERROR	This error indicates that internal error is returned from call control
static int	REDIRECT_CALL_UNKNOWN_PARTY	This error indicates an unknown party is detected, this is an internal error returned from call control
static int	REDIRECT_CALL_UNRECOGNIZED_ MANAGER	This error indicates that internal error is returned from call control
static int	REDIRECT_CALLINFO_ERR	This error indicates that internal error is returned from call control
static int	REDIRECT_ERR	This error indicates that internal error is returned from call control
static int	RETRIEVEFAILED	This error indicates that retrieval of call was rejected by line control or call control
static int	RETRIEVEFAILED_ACTIVE_CALL_ON_LINE	This error indicates that error occurred in retrieving held call; because there is already another active call on the line
static int	SSAPI_NOT_REGISTERED	This error indicates that the redirect command was issued when the internal supporting interface was not initialized; either CTI has not yet finished its initialization or an internal error occurred
static int	TIMEOUT	This error indicates that the request has timed out.
static int	TRANSFER_INACTIVE	This error indicates that attempt to complete transfer, while consult tranfer is not there

Interface	Field	Description
static int	TRANSFERFAILED	This error indicates that the transfer failed probably because one of the call legs was hung up or disconnected from the far end
static int	TRANSFERFAILED_CALLCONTROL_ TIMEOUT	This error indicates that expected response from call control not received during a transfer
static int	TRANSFERFAILED_DESTINATION_BUSY	This error indicates that an attempt is made to transfer call to a busy destination
static int	TRANSFERFAILED_DESTINATION_ UNALLOCATED	This error indicates an attempt is made to to transfer call to a directory number that is not registered
static int	TRANSFERFAILED_OUTSTANDING_ TRANSFER	This error indicates that existing transfer is still in progress
static int	UNDEFINED_LINE	This error indicates that the line that was specified, is not found on the device
static int	UNKNOWN_GLOBAL_CALL_HANDLE	This error indicates that the global call handle is unknown
static int	UNRECOGNIZABLE_PDU	This error indicates that there is a QBE protocol error
static int	UNSPECIFIED	This error indicates that an unspecified error has occured.

Inherited Fields

None

Methods

Table 93: Methods in CiscoJtapiException

Interface	Method	Description
int	getErrorCode()	Returns the errorCode as an integer for this exception.
java.lang.String	getErrorDescription()	Returns the detailed description of the errorCode
java.lang.String	getErrorDescription(int errorCode)	Deprecated Use String getErrorDescription (); instead. Returns the detailed description of the errorCode.
java.lang.String	getErrorName()	Returns an exception in string format.

Interface	Method	Description
java.lang.String	getErrorName(int errorCode)	Deprecated Use String getErrorName (); instead. Returns an exception in string format.

Inherited Methods

None

Related Documentation

See Constant Field Values for more information.

CiscoMediaStreamStartedEv

Applications receive the event when they observe a device that is the target of a “addMediaStream()” invocation. This is the Agent device. This event is sent when the media begins to play on the call.

This event is only deliviered to the device that invokes the original request. Multiple observers on the same address receive the events. Shared lines of the invoking device will not receive this event.

Declaration

public interface CiscoMediaStreamStartedEv extends CiscoCallEv

Fields

None

Inherited Fields

None

Methods

None

Inherited Methods

None

CiscoMediaStreamEndedEv

Applications receive the event when they observe a device that is the target of a “addMediaStream()” invocation. This is the Agent device. This event is sent when the media has finished playing on the call. It contains a field that it exposes if the media is played successfully, or if the end event is the result of an error.

This event is only delivered to the device that invokes the original request. Shared lines of the invoking device will not receive this event.

Declaration

public interface CiscoMediaStreamEndedEv extends CiscoCallEv

Fields

Table 94: Fields in CiscoMediaStreamEndedEv

Interface	Field	Description
static int	RESULT_FAILED	This result code indicates that the CiscoMediaStreamEndedEv is received due to some failure with the request that caused it to end early.
static int	RESULT_SUCCESS	This result code indicates that the CiscoMediaStreamEndedEv is received as a result of successful media streaming.
static int	RESULT_PRIMARY_CALL_DROPPED	This result code indicates that the CiscoMediaStreamEndedEv is received due to the primary call.

Inherited Fields

None

Methods

Table 95: Fields in CiscoMediaStreamEndedEv

Interface	Method	Description
boloean	getResult()	Returns one of the above result codes, which allows the applications to figure out if the CiscoMediaStreamEndedEv is received due to an error, or upon a successful request.

Inherited Methods

None

CiscoJtapiPeer

By extending the com.cisco.services.tracing.TraceModule interface, CiscoJtapiPeer exposes trace information to applications. All instances of JtapiPeer objects that the Cisco JTAPI implementation creates implement this interface. Applications that want to manipulate the trace settings of the Cisco JTAPI implementation may use the CiscoJtapiPeer.getTraceManager method to obtain its TraceManager object. Applications can then manipulate the TraceManager object as described in the com.cisco.services.tracing package.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.
8.5(1)	Added a new API getprovider().

Superinterfaces

CiscoObjectContainer, javax.telephony.JtapiPeer, TraceModule

Declaration

public interface CiscoJtapiPeer extends TraceModule, javax.telephony.JtapiPeer, CiscoObjectContainer

Fields

None

Methods

Table 96: Methods in CiscoJtapiPeer

Interface	Method	Description
CiscoJtapiProperties	getJtapiProperties ()	Defines the various methods that applications can use to modify the parameters that the JTAPI layer will use.
void	setJtapiProperties (CiscoJtapiProperties jtapiproperties)	Provides applications ability to save changes made to CiscoJtapiProperties in jtapi.ini file and activate these changes in properties for the providers in JTAPIPeer.
	getprovider ()	Enhanced to read the singlesignon ticket as ssoticket = "ssoticketfromAD".

Inherited Methods

From Interface com.cisco.services.tracing.TraceModule

getTraceManager, getTraceModuleName

From Interface javax.telephony.JtapiPeer

getName, getProvider, getServices

From Interface com.cisco.jtapi.extensions.CiscoObjectContainer

getObject, setObject

Related Documentation

See CiscoJtapiProperties and TraceModule for more information.

CiscoJtapiPeerImpl

This interface has a method called “getProvider(), ” which is the primary way for applications to open a JTAPI provider. This method takes a “provider string, ” and it was enhanced to take a new argument.

Declaration

public interface CiscoJtapiPeerImpl extends ObjectContainerImpl implements CiscoJtapiPeer

Fields

Methods

Table 97: Methods in CiscoJtapiPeerImpl

Interface	Method	Description
provider	getProvider(String providerString)	The provider string argument is a string of key or value pairs . A new key was added to this method to allow applications to specify whether to run in FIPS compliant mode. The new argument is “FIPSCompliant, ” and applications should specify “true” or “false.” Specifying any value for the FIPSCompliant parameter in the Provider String will have no affect if the provider is not configured as a secured connection.

CiscoJtapiProperties

Cisco Unified JTAPI behavior and functionality is tailored by many parameters which are read in from the jtapi.ini file when an instance of CiscoJtapiPeer is instantiated. These parameters are now exposed to applications for control via this CiscoJtapiproperties interface.

Applications can query the CiscoJtapiproperties properties object and change these parameters to better suit the application functionality. Exposing these properties via the CiscoJtapiproperties interface also allows applications to have a single point of administration (at the application end) for these parameters. The most visible parameters are those describing the tracing levels and tracing destinations.

Usage

```
JtapiPeer peer  =  JtapiPeerFactory.getJtapiPeer ( null );
if(peer instanceof CiscoJtapiPeer)
  {
 CiscoJtapiProperties jProps  =  ((CiscoJtapiPeer)peer).getJtapiProperties();
 jProps.setTracePath("\\D:\\Traces\\WorkFlow");
 jProps.setUseJavaConsoleTrace(false);
 MyProviderObserver providerObserver  =  new MyProviderObserver ();
 provider  =  peer.getProvider ( providerName );
 }
```

In the above example an application has set the Java console tracing to off and set the trace path to D:\Traces\WorkFlowApp1.When the peer is obtained an object implementing CiscoJtapiProperties is created by reading parameters set in the jtapi.ini file. If no jtapi.ini file exists in the classpath default settings are used to create this object. The parameters used by Cisco Jtapi are read in and frozen when the first getProvider () call is made.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.
8.0(1)	Enhanced with methods to enable/disable the HuntList feature.
8.6(1)	Enhanced with methods for applications to specify a desired level of FIPS compliance when they download certificates.

Declaration

public interface CiscoJtapiProperties

Fields

None

Sample Code

```
try
 {
  JtapiPeer peer  =  JtapiPeerFactory.getJtapiPeer ( null );
  MyProviderObserver providerObserver  =  new MyProviderObserver ();
  provider  =  peer.getProvider ( ipaddress;login = useid;passwd = password );
  if ( provider ! =  null )
  {
   provider.addObserver ( providerObserver );
   provInService.waitTrue();
   boolean hlenabled  =
((CiscoJtapiPeer)peer).getJtapiProperties().getHuntListEnabled();
   System.out.Println("Initial state of HuntList  =  " + hlenabled);
   CiscoJtapiProperties cjp  =  ((CiscoJtapiPeer)peer).getJtapiProperties();
   cjp.setHuntListEnabled(true);
   ((CiscoJtapiPeer)peer).setJtapiProperties(cjp);
   boolean hlenabled  =
((CiscoJtapiPeer)peer).getJtapiProperties().getHuntListEnabled();
   System.out.Println("Final state of HuntList  =  " + hlenabled);
  }
```

Methods

Table 98: Methods in CiscoJtapiProperties

Interface	Method	Description
void	deleteCertificates(java.lang.String username, java.lang.String instanceID, java.lang.String ccmCAPFAddress, java.lang.String certificatePath)	Deletes X.509 client certificate installed for USER Instance in certificate store.
void	deleteSecurityPropertyForInstance(java.lang.String username, java.lang.String instanceID, java.lang.String capfIp, java.lang.String certPath)	Deletes security properly from jtapi.ini file and also delete certificate previously installed for username/instanceId.
java.lang.String	getAlarmServiceHostname()	Gets the alarm service host name.
int	getAlarmServicePort()	Gets the port number for the alarm service.
boolean	getCallSecurityStatusChangedEv()	Advises the application if it would receive the event CallSecurityStatusChangedEv when applicable.
int	getCtiRequestTimeout()	Gets the timout for cti requests, other than the provider open (seconds).
java.lang.String[]	getDebuggingNames()	Get names of supported debugging level jtapi traces.
boolean	getDebuggingValue(java.lang.String debuggingName)	Get the enabled or disabled state of a debugging level trace.
int	getDesiredServerHeartbeatInterval()	Get the desired interval at which the CTI Manager must send heartbeats to JTAPI (seconds).

Interface	Method	Description
boolean	getDiscConnBeforeCreatingInCPIC()	Controls the event order for the scenario when only redirected party in observed by Application This interface returns True if ConnDisconnectedEv is send before ConnCreatedEv, False otherwise.
java.lang.String	getFileNameBase()	Gets the filename for individual log files.
java.lang.String	getFileNameExtension()	Gets the filename extension for log files.
Boolean	getHuntListEnabled()	Returns true if HuntList is enabled else false.
int	getJavaSocketConnectTimeout()	Returns the value of service parameter for SOCKET CONNECT TIMEOUT in seconds.
int	getNumTraceFiles()	Gets the number of trace files before rollover.
boolean	getPeriodicWakeupEnabled()	Gets the enabled state of periodic wake up.
int	getPeriodicWakeupInterval()	Gets the interval for periodic wakeup (milliseconds).
boolean	getProcessOfferingAfterNewcallevent()	Retrieves the boolean value for the jtapi.ini parameter ProcessOfferringAfterNewcallEvent'. By default this interface returns false.
int	getProviderOpenRequestTimeout()	Gets the timout for a provider open request (seconds).
int	getProviderOpenRetryAttempts()	Returns the value of service parameter for maximum number of reconnect attempts to CTI Manager.
int	getProviderRetryInterval()	Gets the interval at which the connection to the CTI Manager will ge retried (seconds).
int	getQueueSizeThreshold()	Gets the threshold for the event queue size to trigger alarms.
boolean	getQueueStatsEnabled()	Gets the enabled state of event queue stats.
int	getRouteSelectTimeout()	Gets the route select timeout (milliseconds).
java.util.Hashtable	getSecurityPropertyForInstance()	Returns a Hash table with all the parameters set for users and InstanceIDs. See User/InstanceID Hash Table, on page 193 for key and value pairs.
java.util.Hashtable	getSecurityPropertyForInstance(java.lang.String user, java.lang.String instanceID)	Teturn a Hash table with all the parameters set for users and InstanceIDs. See User/InstanceID Hash Table, on page 193 for key and value pairs.
java.lang.String[]	getServices()	Returns the services that this implementation supports.
java.lang.String	getSyslogCollector()	Gets the syslog collector hostname.

Interface	Method	Description
int	getSyslogCollectorUDPPort()	Gets the syslog collector UDP port.
java.lang.String	getTraceDirectory()	Path directory where trace files will be written.
int	getTraceFileSize()	Trace file size before rollover.
java.lang.String[]	getTraceNames()	Gets the names of supported jtapi traces.
java.lang.String	getTracePath()	Gets the path where the trace files will be located.
boolean	getTraceValue(java.lang.String traceName)	Gets the enabled or disabled state of a trace.
boolean	getUpdateJtapiCalledWithOriginalCalled()	Queries the parameter setting that changes Jtapi behavior on updating called address.
boolean	getUseAlarmService()	gets the enabled/disabled state of the alarm service.
boolean	getUseFileTrace()	Gets the enabled or disabled state of jtapi log file tracing.
boolean	getUseJavaConsoleTrace()	Gets the enabled or disabled state of jtapi console tracing.
boolean	getUseSameDir()	Causes the traces to go to a single directory if UseSameDir is true.
boolean	getUseSyslog()	Gets the enabled or disabled state of syslog tracing.
boolean	IsCertificateUpdated(java.lang.String user, java.lang.String instanceID)	Provides information about where Client and Server certificates are updated for a given user/instanceID or if the Client and Server certificates are not updated.
void	setAlarmServiceHostname(java.lang.String hostname)	Sets the alarm service host name.
void	setAlarmServicePort(int portNumber)	Sets the port number the alarm service is listening on.
void	setCallSecurityStatusChangedEv(boolean val)	Enables applications to set the filter to receive CallSecurityStatusChangedEv to true or false.
void	setCtiRequestTimeout(int seconds)	Sets the time out for CTI requests other than provider open (seconds).
void	setDebuggingValue(java.lang.String debuggingName, boolean value)	Enables or disables a particular debugging level trace.
void	setDesiredServerHeartbeatInterval(int seconds)	Sets the desired interval at which the CTI Manager must send heartbeats to JTAPI (seconds).
void	setDiscConnBeforeCreatingInCPIC(boolean val)	Sets event order, sent Disconnect before Connection created during redirect at redirted party.
void	setFileNameBase(java.lang.String base)	Sets the filename for log files.
void	setFileNameExtension(java.lang.String extn)	Sets the filename extension for log files.

Interface	Method	Description
void	setHuntListEnabled (boolean)	Enables the Hunt List Feature in Cisco Unified JTAPI.
void	setJavaSocketConnectTimeout(int timeout)	Allows application to set the SOCKET CONNECT TIMEOUT in seconds.
void	setNumTraceFiles(int val)	Sets the number of trace files before rollover.
void	setPeriodicWakeupEnabled(boolean enabled)	Sets the enable/disable state for periodic wake up.
void	setPeriodicWakeupInterval(int milliseconds)	Sets the periodic wake up interval (milliseconds).
void	setProcessOfferingAfterNewcallevent(boolean val)	Controls the event order for the transfer scenario when only transfer destination observed by Application and transfer is completed in offering state.
void	setProviderOpenRequestTimeout(int seconds)	Sets the timeout for a provider open request (seconds).
void	setProviderOpenRetryAttempts(int retryAttempts)	Allows application to set the JTAPI Reconnect Attempts to CTI Manager.
void	setProviderRetryInterval(int seconds)	Sets the interval at which the connection to the CTI Manager will ge retried (seconds).
void	setQueueSizeThreshold(int size)	Sets the threshold for the event queue size to trigger alarms.
void	setQueueStatsEnabled(boolean enabled)	Enables and disables event queue statistics.
void	setRouteSelectTimeout(int milliseconds)	Sets the route select timeout milliseconds.
void	setSecurityPropertyForInstance(java.lang.String user, java.lang.String instanceID, java.lang.String authCode, java.lang.String tftp, java.lang.String tftpPort, java.lang.String capf, java.lang.String capfPort, java.lang.String certPath, boolean securityOption)	Deprecated This method is replaced by overloaded method setSecurityPropertyForInstance which takes an extra parameter certStorePassphrase, a passphrase for java key store. This method might have some security vulnerability.
void	setSecurityPropertyForInstance(java.lang.String user, java.lang.String instanceID, java.lang.String authCode, java.lang.String tftp, java.lang.String tftpPort, java.lang.String capf, java.lang.String capfPort, java.lang.String certPath, boolean securityOption, java.lang.String certstorePassphrase)	Provides the application ability to downloading server/client cerfiticate and set security property for application instance in jtapi.ini file of JTAPI.

Interface	Method	Description
void	setSecurityPropertyForInstance(String user, String instanceID, String authCode, String tftp, String tftpPort, String capf, String capfPort, String certPath, boolean securityOption, String certstorePassphrase, boolean fipsCompliant)	This method provides applications the ability to download server/client certificate and set the security property for this application instance in the jtapi.ini file. Specifying any value of fipsCompliant for this method will have no effect unless the securityOption is set to true. It should be noted that this method will call updateCertificate() and updateServerCertificate(). This is the preferred way to acquire certificates.
void	setServices(java.lang.String[] services)	Sets a list of available services.
void	setSyslogCollector(java.lang.String value)	Sets the syslog collector hostname.
void	setSyslogCollectorUDPPort(int port)	Sets the syslog collector UDP port.
void	setTraceDirectory(java.lang.String dir)	Sets the directory where jtapi trace files should be written.
void	setTraceFileSize(int val)	Sets the size of the trace file.
void	setTracePath(java.lang.String path)	Sets the directory root where jtapi traces will be written.
void	setTraceValue(java.lang.String traceName, boolean value)	Enables or disables a particular trace.
void	setUpdateJtapiCalledWithOriginalCalled(boolean val)	Updates Jtapi Called information with original called once the parameter is set to true always.
void	setUseAlarmService(boolean value)	Enables or disables the alarm service.
void	setUseFileTrace(boolean value)	Enables or disables jtapi log file tracing.
void	setUseJavaConsoleTrace(boolean value)	Enables or disables jtapi console tracing.
void	setUseSameDir(boolean value)	Causes the traces to go to a single directory if UseSameDir is true.
void	setUseSyslog(boolean value)	Enables or disables syslog tracing.
void	updateCertificate(java.lang.String user, java.lang.String intanceID, java.lang.String authcode, java.lang.String ccmTFTPAddress, java.lang.String ccmTFTPPort, java.lang.String ccmCAPFAddress, java.lang.String ccmCAPFPort, java.lang.String certificatePath)	Deprecated This method is replaced by overloaded method updateCertifcate which takes an extra parameter certStorePassphrase, a passphrase for java key store. This method might have some security vulnerability.

Interface	Method	Description
void	updateCertificate(java.lang.String user, java.lang.String intanceID, java.lang.String authcode, java.lang.String ccmTFTPAddress, java.lang.String ccmTFTPPort, java.lang.String ccmCAPFAddress, java.lang.String ccmCAPFPort, java.lang.String certificatePath, java.lang.String certStorePassphrase)	Installs an X.509 client certificate for USER Instance in cerfiticate store.
void	updateCertificate(String user, String intanceID, String authcode, String ccmTFTPAddress, String ccmTFTPPort, String ccmCAPFAddress, String ccmCAPFPort, String certificatePath, String certStorePassphrase, boolean fipsCompliant) throws Exception, IOException, UnknownHostException;	This method downloads the client and server certificates for the specified parameters.
void	updateServerCertificate(java.lang.String ccmTFTPAddress, java.lang.String ccmTFTPPort, java.lang.String ccmCAPFAddress, java.lang.String ccmCAPFPort, java.lang.String certificatePath)	Deprecated This method is replaced by overloaded method updateServerCertifcate which takes an extra parameter certStorePassphrase, a passphrase for java key store. This method might have some security vulnerability.
void	updateServerCertificate(java.lang.String userName, java.lang.String instanceID, java.lang.String ccmTFTPAddress, java.lang.String ccmTFTPPort, java.lang.String ccmCAPFAddress, java.lang.String ccmCAPFPort, java.lang.String certificatePath, java.lang.String certStorePassphrase)	Installs an X.509 server certificate given certificate path.
void	updateServerCertificate(String userName, String instanceID, String ccmTFTPAddress, String ccmTFTPPort, String ccmCAPFAddress, String ccmCAPFPort, String certificatePath, String certStorePassphrase, boolean fipsCompliant) throws Exception, IOException, UnknownHostException;	This method downloads the server certificates for the specified parameters. It is called by updateCertificate().

User/InstanceID Hash Table

Table 99: User/InstanceID Hash Table

Key	Value
“user”	userName
String "instanceID"	InstanceID
String"AuthCode"	authCode
String "CAPF"	capfServerIP-Address
String "CAPFPort"	capfServer IP-Address port

Key	Value
String "TFTP"	tftpServer IP-Address
String "TFTPPort"	tftpServer IP-Address port
String "CertPath"	certificate Path
String "securityOption"	Boolean security option (true for enable/ false for disabled)
String "certificateStatus"	Boolean certificate status (true for updated/ false for not updated)

Related Documentation

CiscoLocales

This interface lists all the locales that Cisco Unified JTAPI supports.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoLocales

Fields

Table 100: Fields in CiscoLocales

Interface	Field
static int	LOCALE_ARABIC_ALGERIA
static int	LOCALE_ARABIC_BAHRAIN
static int	LOCALE_ARABIC_EGYPT
static int	LOCALE_ARABIC_IRAQ
static int	LOCALE_ARABIC_JORDAN
static int	LOCALE_ARABIC_KUWAIT
static int	LOCALE_ARABIC_LEBANON

Interface	Field
static int	LOCALE_ARABIC_MOROCCO
static int	LOCALE_ARABIC_OMAN
static int	LOCALE_ARABIC_QATAR
static int	LOCALE_ARABIC_SAUDI_ARABIA
static int	LOCALE_ARABIC_TUNISIA
static int	LOCALE_ARABIC_UNITED_ARAB_EMIRATES
static int	LOCALE_ARABIC_YEMEN
static int	LOCALE_BULGARIAN_BULGARIA
static int	LOCALE_CATALAN_SPAIN
static int	LOCALE_CHINESE_HONG_KONG
static int	LOCALE_CROATIAN_CROATIA
static int	LOCALE_CZECH_CZECH_REPUBLIC
static int	LOCALE_DANISH_DENMARK
static int	LOCALE_DUTCH_NETHERLAND
static int	LOCALE_ENGLISH_UNITED_KINGDOM
static int	LOCALE_ENGLISH_UNITED_STATES
static int	LOCALE_FINNISH_FINLAND
static int	LOCALE_FRENCH_FRANCE
static int	LOCALE_GERMAN_GERMANY
static int	LOCALE_GREEK_GREECE
static int	LOCALE_HEBREW_ISRAEL
static int	LOCALE_HUNGARIAN_HUNGARY
static int	LOCALE_ITALIAN_ITALY
static int	LOCALE_JAPANESE_JAPAN
static int	LOCALE_KOREAN_KOREA
static int	LOCALE_NORWEGIAN_NORWAY
static int	LOCALE_POLISH_POLAND
static int	LOCALE_PORTUGUESE_BRAZIL

Interface	Field
static int	LOCALE_PORTUGUESE_PORTUGAL
static int	LOCALE_ROMANIAN_ROMANIA
static int	LOCALE_RUSSIAN_RUSSIA
static int	LOCALE_SERBIAN_REPUBLIC_OF_MONTENEGRO
static int	LOCALE_SERBIAN_REPUBLIC_OF_SERBIA
static int	LOCALE_SIMPLIFIED_CHINESE_CHINA
static int	LOCALE_SLOVAK_SLOVAKIA
static int	LOCALE_SLOVENIAN_SLOVENIA
static int	LOCALE_SPANISH_SPAIN
static int	LOCALE_SWEDISH_SWEDEN
static int	LOCALE_THAI_THAILAND
static int	LOCALE_TRADITIONAL_CHINESE_CHINA

Methods

None

Related Documentation

See Constant Field Values.

CiscoMasterKeyIndicator

This interface lists the constants for Master Key Indicator.

Table 101: Interface History

Cisco Unified Communications Manager Release Number	Description
10.0(1)	New interface.

Declaration

public interface CiscoMasterKeyIndicator

Methods

Table 102: Methods in CiscoMonitorInitiatorInfo

Interface	Method	Description
static final int	NOT_AVAILABLE	Indicates that MKI (Master Key Indicator) is not present.
static final int	AVAILABLE	Indicates that MKI (Master Key Indicator) is present.

CiscoMediaConnectionMode

The CiscoMediaConnectionMode interface lists all of the media connection modes.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoMediaConnectionMode

Fields

Table 103: Fields in CiscoMediaConnectionMode

Interface	Field	Description
static int	NONE	There is no active transmit or receive channel.
static int	RECEIVE_ONLY	Only the receive channel is active.
static int	TRANSMIT_AND_RECEIVE	Both the transmit and the receive channels are active.
static int	TRANSMIT_ONLY	Only the transmit channel is active.

Methods

None

Related Documentation

See Constant Field Values.

CiscoMediaEncryptionAlgorithmType

The CiscoMediaEncryptionAlgorithmType interface indicates the SRTP algorithm type used for encryption. This interface lists all of the security indicator values that an application can get in CiscoRTPInputKeyEv and CiscoRTPOutputKeyEv. If an application is terminating its own media on CTIPorts and Media Terminated RPs, only one of the following algorithms needs to be provided in the register API.

Interface History

Cisco Unified Communications Manager Release	Description
3.x	Added the extension.

Superinterfaces

public interface CiscoMediaEncryptionAlgorithmType

Fields

Table 104: Fields in CiscoMediaEncryptionAlgorithmType

Inteface	Field	Description
staticint	AES_128_COUNTER	The algorithm used is based on Advanced Encryption Standard (AES), which is a computer security standard. The cryptography scheme is a symmetric block cipher that encrypts and decrypts 128-bit blocks of data.

Related Documentation

See Constant Field Values for additional information.

CiscoMediaEncryptionKeyInfo

The CiscoMediaEncryptionKeyInfo interface lets applications get information about SRTP keys.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoMediaEncryptionKeyInfo

Fields

None

Methods

Table 105: Methods in CiscoMediaEncryptionKeyInfo

Interface	Method	Description
int	getAlgorithmID()	Returns the media encryption algorithm ID for the current stream.
int	getIsMKIPresent()	Indicates whether MKI is present.
byte[]	getKey()	Returns the master key for the stream.
int	getKeyLength()	Returns the keyLength of the key.
byte[]	getSalt()	Returns the salt key for the stream.
int	getSaltLength()	Returns the saltLength of the salt.
int	keyDerivationRate()	Indicates the SRTP key derivation rate for this session.

Related Documentation

See CiscoRTPInputKeyEv, CiscoRTPOutputKeyEv.

CiscoMediaOpenIPPortEv

CiscoMediaOpenIPPortEv event is delivered only if the terminal is registered with registration type as CiscoBaseTerminal.DYNAMIC_MEDIA_REGISTRATION_FOR_GET_PORT_SUPPORT or CiscoBaseTerminal.STATIC_MEDIA_REGISTRATION_FOR_GET_PORT_SUPPORT.

Interface History

Cisco Unified Communications Manager Release Number	Description
8.5(1)	New interface.

Sample Code

```
public class MyTermObserver implements implements TerminalObserver,
    CallControlTerminalObserver,  AgentTerminalObserver,  PhoneTerminalObserver
 {
  public void termChangedEvent (TermEv[] evlist)
   {
   for(int i = 0; evlist ! =  null && i < evlist.length; i++)
     {
      …
      …
      If ( evlisth[i] instanceof  CiscoMediaOpenIPPortEv)
        {
        CiscoMediaOpenIPPortEv ev  =  (CiscoMediaOpenIPPortEv)evlist[i];
        if(((CiscoBaseMediaTerminal)(ev.getTerminal())))).getRegistrationType
             =  =  CiscoTerminal.DYNAMIC_MEDIA_REGISTRATION_FOR_GET_PORT_SUPPORT)

          {
          System.out.println("Set RTP parameters");
          System.out.println("open the port");
          } else {
          System.out.println("Open port");
          }
        }
      }
        …
        …
  }
```

Declaration

public interface CiscoMediaOpenIPPortEv

Superinterfaces

NA

Fields

NA

Inherited Fields

NA

Methods

Table 106: Methods in CiscoMediaOpenIPPortEv

Interface	Method
int	getMediaIPAddressingMode()
CiscoRTPHandle	getCiscoRTPHandle()

Inherited Methods

NA

CiscoMediaOpenLogicalChannelEv

The system sends a CiscoMediaOpenLogicalChannelEv event each time that media gets established for a dynamically registered CiscoMediaTerminal or CiscoRouteTerminal. Upon receiving this event, applications must invoke setRTPParams on CiscoMediaTerminal or CiscoRouteTerminal and pass in the IP address and port number where they want to terminate the media, along with the rtpHandle that this event delivers.

Applications can get a call reference by using CiscoProvider.getCall(CiscoRTPHandle). Applications must be aware that the far end and local end may not be able to invoke features unless the setRTPParams method is invoked. If applications fail to respond to this event within the specified time, the call may get disconnected.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.0(1)	Added the getAddressingModeForMedia() method.
8.5(1)	Added the isRTPRequired() method.

Sample Code

```
public class MyTermObserver implements implements TerminalObserver,
       CallControlTerminalObserver,  AgentTerminalObserver,  PhoneTerminalObserver

 {
  public void termChangedEvent (TermEv[] evlist)
   {
   for(int i = 0; evlist ! =  null && i < evlist.length; i++){
     …
     …
     If ( evlisth[i] instanceof  CiscoMediaOpenLogicalChannelEv)
      {
      CiscoMediaOpenLogicalChannelEv ev  =
(CiscoMediaOpenLogicalChannelEv)evlist[i];
      if(ev.isRTPRequired())
      {
      System.out.println("Set RTP parameters");
```

```
      } else {
      System.out.println("Do not set RTP parameters");
      }
   }
 }
     …
     …
 }
```

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoMediaOpenLogicalChannelEv extends CiscoTermEv

Fields

Table 107: Fields in CiscoMediaOpenLogicalChannelEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 108: Methods in CiscoMediaOpenLogicalChannelEv

Interface	Method	Description
int	getAddressingModeForMedia()	Returns int Application and could get following value for required IP Addressing Mode: • CiscoTerminal.IP_ADDRESSING_IPv4—Means application needs to provide IPv4 format for the IP Address in setRTPParams request. • CiscoTerminal.IP_ADDRESSING_IPv6: Means application need to provide IPv6 format IP Address in set RTP Params request.
CiscoRTPHandle	getCiscoRTPHandle()	Returns CiscoRTPHandle object. Applications should pass this handle along with RTPParameters to CiscoMediaTerminal or CiscoRouteTerminal. Applications can get call reference using CiscoProvider.getCall If there is no callobserver or there was no callobserver when this event is delivered, then CiscoProvider.getCall may return null
int	getMediaConnectionMode()	Returns a CiscoMediaConnectionMode. Applications could get one of the following values: • CiscoMediaConnectionMode. RECEIVE_ONLY—Means one-way media receive only. • CiscoMediaConnectionMode. TRANSMIT_AND_RECEIVE—Means two-way media. Applications should never see a value of NONE; however, if that happens, applications should ignore the event and log an error.
int	getPacketSize()	Returns the packet size of the far end, in milliseconds. getPacketSize

Interface	Method	Description
int	getPayLoadType()	Returns the payload format of the far end, one of the following constants: • CiscoRTPPayload.NONSTANDARD • CiscoRTPPayload.G711ALAW64K • CiscoRTPPayload.G711ALAW56K • CiscoRTPPayload.G711ULAW64K • CiscoRTPPayload.G711ULAW56K • CiscoRTPPayload.G722_64K • CiscoRTPPayload.G722_56K • CiscoRTPPayload.G722_48K • CiscoRTPPayload.G7231 • CiscoRTPPayload.G728 • CiscoRTPPayload.G729 • CiscoRTPPayload.G729ANNEXA • CiscoRTPPayload.IS11172AUDIOCAP • CiscoRTPPayload.IS13818AUDIOCAP • CiscoRTPPayload.ACY_G729AASSN • CiscoRTPPayload.DATA64 • CiscoRTPPayload.DATA56 • CiscoRTPPayload.GSM • CiscoRTPPayload.ACTIVEVOICE
boolean	isRTPRequired()	Indicates if the application must set the RTP parameters upon receiving this event.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values and CiscoRTPParams

CiscoMediaSecurityIndicator

CiscoMediaSecurityIndicator gets sent in CiscoRTPInputKeyEv, CiscoRTPOutputKeyEv, and CiscoSnapShotRTPEv. It shows the call security status.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoMediaSecurityIndicator

Fields

Table 109: Fields in CiscoMediaSecurityIndicator

Interface	Field	Description
staticint	MEDIA_ENCRYPT_KEYS_AVAILABLE	Terminates the media in secure mode, and keys are available.
staticint	MEDIA_ENCRYPT_KEYS_UNAVAILABLE	Terminates the media in secure mode, but keys are not available because SRTP is not enabled in Cisco Unified Communications Manager Administration.
staticint	MEDIA_ENCRYPT_USER_NOT_AUTHORIZED	Terminates the media in secure mode, but keys are not available because the user is not authorized to get the keys.
staticint	MEDIA_NOT_ENCRYPTED	The media is not encrypted for this call.

Related Documentation

See Constant Field Values.

CiscoMediaTerminal

A CiscoMediaTerminal is a special kind of CiscoTerminal that allows applications to terminate RTP media streams. Unlike a CiscoTerminal, a CiscoMediaTerminal does not represent a physical telephony endpoint, which is observable and controllable in a third-party manner. Instead, a CiscoMediaTerminal is a logical telephony endpoint, which may be associated with any application that wants to terminate media. Such applications include voice messaging systems, interactive voice response (IVR), and softphones.

Note Only CTIPorts appear as CiscoMediaTerminals through Cisco Unified JTAPI.

Terminating media is a two-step process. To terminate media for a particular terminal, an application first adds an observer that implements the CiscoTerminalObserver interface using the Terminal.addObserver method. Finally, the application registers the IP address and port number to which incoming RTP streams for the terminal should be directed, by using the CiscoMediaTerminal.register method.

To supply an IP address and port number dynamically on a per-call basis, applications must register by only providing the capabilities that they support. Applications must react to the CiscoMediaOpenLogicalChannelEv that gets sent whenever media gets established. Applications registering with this type must be aware that, when this event is received, the far end and the local end may not be able to perform any feature operation unless media is established. If applications fail to respond to this event within the specified time, the call may get dropped.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1x	Support added for IPv6.

Superinterfaces

CiscoObjectContainer, CiscoTerminal, javax.telephony.Terminal

Declaration

public interface CiscoMediaTerminal extends CiscoTerminal

Fields

None

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoTerminal

ASCII_ENCODING, DEVICESTATE_ACTIVE, DEVICESTATE_ALERTING, DEVICESTATE_HELD, DEVICESTATE_IDLE, DEVICESTATE_UNKNOWN, DEVICESTATE_WHISPER,

DND_OPTION_CALL_REJECT, DND_OPTION_NONE, DND_OPTION_RINGER_OFF, IN_SERVICE, IP_ADDRESSING_MODE_IPV4, IP_ADDRESSING_MODE_IPV4_V6, IP_ADDRESSING_MODE_IPV6, IP_ADDRESSING_MODE_UNKNOWN, IP_ADDRESSING_MODE_UNKNOWN_ANATRED, NOT_APPLICABLE, OUT_OF_SERVICE, UCS2UNICODE_ENCODING, UNKNOWN_ENCODING

Methods

Table 110: Methods in CiscoMediaTerminal

Interface	Method	Description
void	register(java.net.InetAddressaddress, intport, CiscoMediaCapability[]capabilities)	This method registers the MediaTerminal and returns successfully when the MediaTerminal is registered. The CiscoMediaTerminal must be in the CiscoTerminal.UNREGISTERED state and its Provider must be in the Provider.IN_SERVICE state. This method has three arguments: • The first argument specifies the internet address at which the RTP media stream for this Terminal will terminate. • The second indicates the UDP port at which RTP packets will be directed. • The final argument indicates the type of RTP encodings that the application is willing to support for this Terminal. **Parameters** • address—The internet address at which inbound IPv4 RTP streams on this terminal will terminate • port—The UDP port for inbound RTP streams on this terminal • capabilities—The list of the types of RTP encodings that the application supports for this terminal. **Throws** CiscoRegistrationException

Interface	Method	Description
void	register(java.net.InetAddressaddress, intport)	Deprecated Registers a Terminal with the specified address and port, defaulting to G.711 64 kHz u-law encoding with a thirty-millisecond packet size. **Parameters** • address—The internet address for inbound IPv4 RTP streams on this terminal • port—The UDP port for inbound RTP streams on this terminal **Throws** CiscoRegistrationException
void	register(java.net.InetAddressaddress, intport, CiscoMediaCapability[]capabilities, int[]algorithmIDs)	This method registers the MediaTerminal. Ensure that the CiscoMediaTerminal is in the CiscoTerminal.UNREGISTERED state and its Provider is in the Provider.IN_SERVICE state. This method returns successfully when the MediaTerminal gets registered. This method requires that the application have a TLS link established with CTIManager and have the SRTP Enabled flag enabled in Cisco Unified Communications Manager Administration for the user; otherwise, the system throws a PrivilegeViolationException. **Parameters** • address—The internet address for inbound IPv4 RTP streams on this terminal • port—The UDP port for inbound RTP streams on this terminal • capabilities—The list of RTP encodings that this terminal supports • algorithmIDs—The SRTP algorithms that this CTIPort supports. AlgorithmIDs must be one of CiscoMediaEncryptionAlgorithmType. **Throws** CiscoRegistrationException javax.telephony.PrivilegeViolationException

Interface	Method	Description
void	register(java.net.InetAddressaddress, intport, CiscoMediaCapability[]capabilities, int[]algorithmIDs, java.net.InetAddressaddress_v6, intactiveAddressingMode)	

Interface	Method	Description
		The CiscoMediaTerminal must be in the CiscoTerminal.UNREGISTERED state and its Provider must be in the Provider.IN_SERVICE state. The successful effect of this method is to register the MediaTerminal. The activeAddressingMode indicates the application IP addressing capabilities. If application specifies activeAddressingMode as CiscoTerminal.IP_ADDRESSING_MODE_IPv4, then it must also specify address. If application specifies activeAddressingMode as CiscoTerminal.IP_ADDRESSING_MODE_IPv6, then it must also specify address_v6. If application specifies activeAddressingMode as CiscoTerminal.IP_ADDRESSING_MODE_IPv4_6, then it must also specify address and address_v6. **Method Arguments** This method has four arguments: • The first argument specifies the internet address at which the RTP media stream for this Terminal will be terminated. • The second indicates the UDP port at which RTP packets will be directed. • The third argument indicates the type of RTP encodings that the application is willing to support for this Terminal • The final argument indicates SRTP algorithm that application supports. This method can be used only if application has TLS link established with CTIManager and if application has SRTP Enabled flag enabled in CM Admin pages for the user, otherwise PrivilegeViolationException is thrown. **Method Post-conditions** This method returns successfully when the MediaTerminal is registered. **Parameters** • address—The internet address for inbound IPv4 RTP streams on this terminal, it can be null depending on application Addressing Mode. • port—The UDP port for inbound RTP streams on this terminal

Interface	Method	Description
		• capabilities—The list of RTP encodings supported by this terminal • algorithmIDs—Indicates SRTP algorithms that this CTIPort supports. AlgorithmIDs may only be one of CiscoMediaEncryptionAlgorithmType • address_v6—The IPv6 internet address for inbound IPv6 RTP streams on this terminal, it can be null depending upon activeAddressingMode • activeAddressingMode—IP Addressing mode in which application intends to register this CiscoMediaTerminal. It can be: CiscoTerminal.IP_ADDRESSING_MODE_IPv4 CiscoTerminal.IP_ADDRESSING_MODE_IPv6 CiscoTerminal.IP_ADDRESSING_MODE_IPv4z_v6 **Since: 7.0** **Throws** CiscoRegistrationException, javax.telephony.PrivilegeViolationException

Interface	Method	Description
void	register(CiscoMediaCapability[]capabilities)	This method registers the MediaTerminal with the specified CiscoMediaCapabilities. Applications should use this method when they want to supply the IP address and port dynamically for each call. Applications that register with this method will receive a CiscoMediaOpenLogicalChannelEv for each call and must supply an IP address and port number by using the setRTPParams method on this object. Ensure the CiscoMediaTerminal is in the CiscoTerminal.UNREGISTERED state and its Provider is in the Provider.IN_SERVICE state. **Method Arguments** Arguments indicate the type of RTP encodings that the application is willing to support for this Terminal. **Method Post-conditions** This method returns successfully when the CiscoMediaTerminal is registered. **Parameters** capabilities—The list of RTP encodings that this terminal supports. **Throws** CiscoRegistrationException

Interface	Method	Description
void	register(CiscoMediaCapability[]capabilities, int[]algorithmIDs)	This method registers a MediaTerminal with the specified CiscoMediaCapabilities and supported SRTP algorithms. Applications should use this method when they want to supply the IP address and port dynamically for each call and also want to specify the SRTP algorithm. Applications that register with this method will receive a CiscoMediaOpenLogicalChannelEv for each call and must supply the IP address and port number by using the setRTPParams method on this object. This form of register() also requires a second parameter that indicates which SRTP algorithm that the application supports. This method requires that the application have a TLS link established with CTIManager and have the SRTP Enabled flag enabled in Cisco Unified Communications Manager Administration for the user; otherwise, the system throws a PrivilegeViolationException. This method returns successfully when the CiscoMediaTerminal gets registered. Ensure the CiscoMediaTerminal is in the CiscoTerminal.UNREGISTERED state and its Provider is in the Provider.IN_SERVICE state. **Parameters** • capabilities—The list of RTP encodings that this terminal supports • algorithmIDs—The list of SRTP algorithms that this terminal supports. AlgorithmIDs must be one of CiscoMediaEncryptionAlgorithmType. **Throws** CiscoRegistrationException javax.telephony.PrivilegeViolationException

Interface	Method	Description
void	register(CiscoMediaCapability[]capabilities, int[]algorithmIDs, intactiveAddressingMode)	

Interface	Method	Description
		The CiscoMediaTerminal must be in the CiscoTerminal.UNREGISTERED state and its Provider must be in the Provider.IN_SERVICE state. The successful effect of this method is to register the MediaTerminal. It registers a Terminal with specified CiscoMediaCapabilities and supported SRTP algorithms. It also indicates that application is interested in supplying ipAddress and port dynamically for each call. Applications registering with this method receive CiscoMediaOpenLogicalChannelEv for each call and have to supply ipAddress and port number using setRTPParams method on this object. The second parameter indicates SRTP algorithm that application supports. This method can be used only if application has TLS link established with CTIManager and if application has SRTP Enabled flag enabled in Cisco Unified Communications Manager Administration for the user, otherwise PrivilegeViolationException is thrown. **Method Arguments** Arguments indicate the type of RTP encodings that the application is willing to support for this Terminal and the application or CTIManager failure persistence delay. **Method Post-conditions** This method returns successfully when the CiscoMediaTerminal is registered. **Parameters** • capabilities—The list of RTP encodings supported by this terminal • algorithmIDs—Indicates the list of SRTP algorithms supported by this terminal. AlgorithmIDs may only be one of CiscoMediaEncryptionAlgorithmType • activeAddressingMode—Is the IP Addressing mode in which application intends to register this CiscoMediaTerminal. The activeAddressingMode can be: CiscoTerminal.IP_ADDRESSING_MODE_IPv4 CiscoTerminal.IP_ADDRESSING_MODE_IPv6 CiscoTerminal.IP_ADDRESSING_MODE_IPv4_v6 **Throws**

Interface	Method	Description
		CiscoRegistrationException javax.telephony.PrivilegeViolationException
void	setRTPParams(CiscoRTPHandlertpHandle, CiscoRTPParamsrtpParams)	Applications must use this method when they want to set the IP address and RTP port number to dynamically stream media for a call. In this situation, the application will have registered the MediaTerminal or CiscoRouteTeminal by providing only capabilities. Applications must invoke this method upon receiving the CiscoCallOpenLogicalChannel on terminalObserver. Applications must pass in the rtpHandle that they receive in CiscoCallOpenLogicalChannelEv. Applications can get a CiscoCall reference by calling the CiscoProvider.getRTPHandle(rtpHandle) method. This method may return null if no call observer is added on the terminal, or there was no callobserver at the time when this event got sent sent, or or there is no call associated with this handle. **Parameters** • rtpHandle—is obtained from. CiscoMediaCallOpenLogicalChannelEv • rtpParams—is of type CiscoRTPParams, which is used to specify the dynamic RTP address and port number for a media terminal on a per-call basis. **Throws** javax.telephony.InvalidStateException javax.telephony.InvalidArgumentException javax.telephony.PrivilegeViolationException
void	unregister()	This method unregisters the MediaTerminal and returns successfully when the MediaTerminal gets unregistered. The CiscoMediaTerminal must be registered and its Provider must be in the Provider.IN_SERVICE state. **Throws** CiscoUnregistrationException
boolean	isRegistered()	This method returns true if the CiscoMediaTerminal is registered and false otherwise. For a MediaTerminal, this method returns true if the MediaTerminal is InService and false if it is OutOfService. For CTIManager failure cases, this method returns false.

Interface	Method	Description
boolean	isRegisteredByThisApp()	This method returns true if this application issued a successful registration request. The registration remains valid even if the device is out-of-service because of a CTIManager failure. This will get set to true until this application unregisters the device.
int	getIPAddressingMode()	An application can invoke this API to query the IP Addressing Mode of the CiscoMediaTerminal Addressing mode may be any of the following constants: • CiscoTerminal.IP_ADDRESSING_IPv4 • CiscoTerminal.IP_ADDRESSING_IPv6 • CiscoTerminal.IP_ADDRESSING_IPv4_v6

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoTerminal

createSnapshot, getAltScript, getDeviceState, getDNDOption, getDNDStatus, getEMLoginUsername, getFilter, getLocale, getProtocol, getRegistrationState, getRTPInputProperties, getRTPOutputProperties, getState, getSupportedEncoding, isRestricted, sendData, sendData, setDNDStatus, setFilter, unPark

From Interface javax.telephony.Terminal

addCallObserver, addObserver, getAddresses, getCallObservers, getCapabilities, getName, getObservers, getProvider, getTerminalCapabilities, getTerminalConnections, removeCallObserver, removeObserver

From Interface com.cisco.jtapi.extensions.CiscoObjectContainer

getObject, setObject

Related Documentation

See CiscoTerminal and CiscoMediaOpenLogicalChannelEv.CiscoRTPParams

CiscoMonitorInitiatorInfo

This interface defines provides information about the monitor initiator.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoMonitorInitiatorInfo

Fields

None

Methods

Table 111: Methods in CiscoMonitorInitiatorInfo

Interface	Method	Description
CiscoAddress	getAddress()	Returns the monitor initiator address.
Int	getMonitorInitiatorCallLegHandle()	Returns the call leg hanlde at the monitor initiator. JTAPI gets the call at the monitor target by using provider.getCall(int monitorInitiatorCallLegHandle). This method returns null if the call at the monitor initiator is not active in this provider.
java.lang.String	getTerminalName()	Returns the terminal name of the monitor initiator.

Related Documentation

None

CiscoMonitorTargetInfo

This interface provides information about the monitor target.

Declaration

public interface CiscoMonitorTargetInfo

Fields

None

Methods

Table 112: Methods in CiscoMonitorTargetInfo

Interface	Method	Description
CiscoAddress	getAddress()	Returns the monitor target address.
Int	getMonitorTargetCallLegHandle()	Returns the call leg handle at the monitor target.
java.lang.String	getTerminalName()	Returns the terminal name of monitor target.

Related Documentation

None

CiscoMultiForkingRecorderInfo

The CiscoMultiForkingRecorderInfo interface contains the information related to the recorders.

Interface History

Cisco Unified Communications Manager Release Number	Description
12.5(1)	A new API which returns recorders details during Multi Forking Recording. `CiscoMultiForkingRecorderInfo()` New constants: CALL_RECORDING_MULTI_FORKING _RECORDER_TYPE_UNKNOWN CALL_RECORDING_MULTI_FORKING _RECORDER_TYPE_OPTIONAL_RECORDER CALL_RECORDING_MULTI_FORKING _RECORDER_TYPE_MANDATORY_RECORDER CALL_RECORDING_MULTI_FORKING_RECORDER_STATUS_UNKNOWN CALL_RECORDING_MULTI_FORKING_RECORDER_STATUS_SUCCESS CALL_RECORDING_MULTI_FORKING_RECORDER_STATUS_FAILURE

Declaration

public interface CiscoMultiForkingRecorderInfo

Methods

Methods in CiscoMultiForkingRecorderInfo

Interface	Method	Description
java.lang.String	getRecorderURI()	This interface returns URI of the MultiForking recorder.
java.lang.String	getRecorderErrorMsg()	This interface returns the error message of the MultiForking recorder.
public int	getRecorderType()	This interface returns the integer which denotes the type of recorder. The recorder type can be: CALL_RECORDING_MULTI_FORKING _RECORDER_TYPE_UNKNOWN CALL_RECORDING_MULTI_FORKING _RECORDER_TYPE_OPTIONAL_RECORDER CALL_RECORDING_MULTI_FORKING _RECORDER_TYPE_MANDATORY_RECORDER
public int	get RecorderStatus()	This interface returns the integer which denotes the status of the recorder. The recorder status can be: CALL_RECORDING_MULTI_FORKING _RECORDER_STATUS_UNKNOWN CALL_RECORDING_MULTI_FORKING _RECORDER_STATUS_SUCCESS CALL_RECORDING_MULTI_FORKING _RECORDER_STATUS_FAILURE

CiscoMultiMediaCapabilityInfo

CiscoMultiMediaCapabilityInfo interface contains the multimedia capabilities of a terminal. Applications can get the video capability, telepresence interopability, and number of screens information of the terminal using this API.

Declaration

public interface CiscoMultiMediaCapabilityInfo

com.cisco.jtapi.extensions.CiscoMultiMediaCapabilityInfo

Fields

Table 113: Fields in CiscoMultiMediaCapabilityInfo

Interface	Field	Description
static final int	VIDEO_DISABLED	Indicates that the CiscoMultiMediaCapabilityInfo. getVideoCapability () for this terminal is CiscoMultiMediaCapabilityInfo.ENABLED.
static final int	VIDEO_ENABLED	Indicates that the CiscoMultiMediaCapabilityInfo. getVideoCapability () for this terminal is CiscoMultiMediaCapabilityInfo.ENABLED.
static final int	TELEPRESENCEINTEROP_NONE	Indicates that the CiscoMultiMediaCapabilityInfo. getTelepresenceInfo() for this terminal is TELEPRESENCEINTEROP_NONE.
static final int	TELEPRESENCEINTEROP_ENABLED	Indicates that the CiscoMultiMediaCapabilityInfo. getTelepresenceInfo () for this terminal is CiscoMultiMediaCapabilityInfo. TELEPRESENCEINTEROP_ENABLED.
static final int	UNKNOWN	This field will return -1 to indicate that the video capability and telepresence interopability is screen count is Unknown.

Methods

Table 114: Methods in MultiMediaCapabilityInfo

Interface	Method	Description
int	getVideoCapability()	Returns the video capability of the Terminal. The video capability can be CiscoMultiMediaCapabilityInfo. DISABLED or CiscoMultiMediaCapabilityInfo. ENABLED.
int	getTelepresenceInfo()	Returns the telepresence interoperability of the Terminal. The telepresence interoperability can be CiscoMultiMediaCapabilityInfo. TELEPRESENCEINTEROP_DISABLED or CiscoMultiMediaCapabilityInfo. TELEPRESENCEINTEROP_ENABLED.
int	getScreenCount()	Returns the number of screens present on the Terminal.

CiscoMultiMediaConnectionMode

This interface specifies the connection mode associated with the multimedia stream.

Table 115: Interface History

Cisco Unified Communications Manager Release Number	Description
10.0(1)	New interface.

Declaration

public interface CiscoMultiMediaConnectionMode

Methods

Table 116: Methods in CiscoProvTerminalMultiMediaConnectionMode

Interface	Field	Description
static final int	RECEIVE_ONLY	Indicates that only the receive channel is active.
static final int	TRANSMIT_ONLY	Indicates that only the transmit channel is active.

CiscoMultiMediaEncryptionKeyInfo

This interface contains the multi media streams encryption key information of a Terminal.

Table 117: Interface History

Cisco Unified Communications Manager Release Number	Description
10.0(1)	New interface.

Declaration

public interface CiscoMultiMediaEncryptionKeyInfo

Methods

Table 118: Methods in CiscoProvTerminalMultiMediaEncryptionKeyInfo

Interface	Method	Description
byte[]	getRxKey()	Returns the master key for the input stream.
byte[]	getRxSalt()	Returns the salt key for the input stream.
byte[]	getTxKey()	Returns the master key for the output stream.
byte[]	getTxSalt()	Returns the salt key for the output stream.
int	getAlgorithmID()	Returns the media encryption algorithm ID for the current stream.
int	getRxMKIPresent()	Indicates whether MKI is present for the input stream.
int	getTxMKIPresent()	Indicates whether MKI is present for the output stream.

CiscoMultiMediaProperties

This interface contains the multi media stream information of the video-capabilities on a Terminal.

Table 119: Interface History

Cisco Unified Communications Manager Release Number	Description
10.0(1)	New interface.

Declaration

public interface CiscoMultiMediaProperties

Methods

Table 120: Methods in CiscoProvTerminalMultiMediaProperties

Interface	Method	Description
CiscoRTPProperties	CiscoRTPProperties	Returns the rtp properties for the multimedia stream.

Interface	Method	Description
int	getMultiMediaConnection Mode()	Returns the connection mode for the multimedia stream. The multimedia connection mode can be: • CiscoMultiMediaConnectionMode. INACTIVE = 3 • CiscoMultiMediaConnectionMode. RECEIVE_ONLY = 1; • CiscoMultiMediaConnectionMode. TRANSMIT_ONLY = 2 • CiscoMultiMediaConnectionMode. TRANSMIT_AND_RECEIVE = 0
int	getMultiMediaType()	Returns the multimedia type for the multimedia stream. The media type can be • CiscoMultiMediaType. INVALID = 0 • CiscoMultiMediaType. AUDIO = 1 • CiscoMultiMediaType. MAIN_VIDEO = 2 • CiscoMultiMediaType. PRESENTATION_VIDEO = 3
boolean	isKeyInfoPresent()	Returns whether key information is present for the multimedia stream.
CiscoMultiMedia EncryptionKeyInfo	getMultiMediaEncryption KeyInfo()	Returns the multimedia encryption data for the multimedia stream.
int	getMultiMediaSecurity Indicator()	Indicates security indicator for the multimedia stream. The security indicator can be: • CiscoMasterKeyIndicator. NOT_AVAILABLE = 0 • CiscoMasterKeyIndicator. AVAILABLE = 1

CiscoMultiMediaStreamsInfoEv

CiscoMultiMediaStreamsInfoEv is a new interface that is being notified to applications as a Terminal Event. This interface will be delivered to terminal observers added by applications to indicate the multimedia streams information.

Table 121: Interface History

Cisco Unified Communications Manager Release Number	Description
10.0(1)	New interface.

Declaration

public interface CiscoMultiMediaStreamsInfoEv extends CiscoTermEv

Methods

Table 122: Methods in CiscoProvTerminalMultiMediaStreamsInfoEv

Interface	Field	Description
CiscoMultiMediaProperties[]	getProperties()	Returns an array of CiscoMultiMediaProperties, one for each stream.
CiscoCallID	getCallID()	Returns CiscoCallID.

CiscoMultiMediaType

This interface specifies the multimedia type associated with the multimedia stream.

Table 123: Interface History

Cisco Unified Communications Manager Release Number	Description
10.0(1)	New interface.

Declaration

public interface CiscoMultiMediaType

Methods

Table 124: Methods in CiscoProvTerminalMultiMediaType

Interface	Field	Description
static final int	INVALID	The multimedia stream type is unknown.

Interface	Field	Description
static final int	AUDIO	The multimedia stream contains audio information.
static final int	MAIN_VIDEO	The multimedia stream contains video information.
static final int	PRESENTATION_VIDEO	The multimedia stream contains presentation information.

CiscoObjectContainer

The ApplicationObject interface allows applications to associate an application-defined object to objects that implement this interface.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Subinterfaces

CiscoAddress, CiscoCall, CiscoCallID, CiscoConnection, CiscoConnectionID, CiscoConsultCall, CiscoIntercomAddress, CiscoJtapiPeer, CiscoMediaTerminal, CiscoProvider, CiscoRouteTerminal, CiscoTerminal, CiscoTerminalConnection

Declaration

public interface CiscoObjectContainer

Fields

None

Methods

Table 125: Methods in CiscoObjectContainer

Interface	Method	Description
java.lang.Object	getObject()	Gets the application-defined object.
java.lang.Object	setObject(java.lang.Objectreference)	Sets an application-defined object.

Related Documentation

None

CiscoOutOfServiceEv

The CiscoOutOfServiceEv event is the super class for the out-of-service events CiscoAddrOutOfServiceEv and CiscoTermOutOfServiceEv. This class defines the causes for out-of-service events.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, javax.telephony.events.Ev

Subinterfaces

CiscoAddrOutOfServiceEv, CiscoTermOutOfServiceEv

Declaration

public interface CiscoOutOfServiceEv extends CiscoEv

Fields

Table 126: Fields in CiscoOutOfServiceEv

Interface	Field	Description
staticint	CAUSE_CALLMANAGER_FAILURE	The cause for this event is due a Cisco Unified Communications Manager failure.
staticint	CAUSE_CTIMANAGER_FAILURE	The cause for this event is due to a failure from CTIManager.
staticint	CAUSE_DEVICE_FAILURE	The cause for this event is a device failure.
staticint	CAUSE_DEVICE_RESTRICTED	The cause for this event is that the device is restricted.
staticint	CAUSE_DEVICE_UNREGISTERED	The cause for this event is that the device is in an unregistered state.

Interface	Field	Description
staticint	CAUSE_LINE_RESTRICTED	The cause for this event is that the line is restricted.
staticint	CAUSE_NOCALLMANAGER_AVAILABLE	The cause for this event is the unavailability of any Cisco Unified Communications Manager.
staticint	CAUSE_REHOME_TO_HIGHER_PRIORITY_CM	The cause for this event is an to error in failback to a higher-priority Cisco Unified Communications Manager node.
staticint	CAUSE_REHOMING_FAILURE	The cause for this event is a failure while attempting to rehome.
staticint	ID	—

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Methods

None

Related Documentation

See Constant Field Values

CiscoPartyInfo

This interface defines the party info of the call.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoPartyInfo

Fields

Table 127: Fields in CiscoPartyInfo

Interface	Field	Description
Static int	ABBREVIATED_NUMBER	This NumberType is same as 4; it represents caller is from same Cisco Unified Communications Manager server.
Static int	INTERNATIONAL_NUMBER	This NumberType is same as 0; it represents nothing is configured
Static int	NATIONAL_NUMBER	This NumberType is same as 1; it represents caller is INTERNATIONAL
Static int	NET_SPECIFIC_NUMBER	This NumberType is same as 2; it represents caller is NATIONAL
Static int	RESERVED_FOR_EXTENSION	This NumberType is same as 6; it represents its a fast dial call - not being used currently
Static int	SUBSCRIBER_NUMBER	This NumberType is same as 3; it represents call is from MGCP/H.323 gateway
Static int	UNKNOWN_NUMBER	—

Methods

Table 128: Methods in CiscoPartyInfo

Interface	Method	Description
javax.telephony.Address	getAddress()	Returns the address.
boolean	getAddressPI()	Returns Presentation Indicator (PI) associated with Address. If it returns true, Application can display this Address name to end users. If it returns false, Applications should not display this Address name to end user.
java.lang.String	getDisplayName()	Returns display name of the party.

Interface	Method	Description
boolean	getDisplayNamePI()	Returns the PI associated with DisplayName. If it returns true, Application can display this DisplayName to end users. If it returns false, Applications should not display this DisplayName to end user.
int	getlocale()	Returns the locale of the party unicode display name.
int	getNumberType()	Returns number type of the party.
java.lang.String	getUnicodeDisplayName()	Returns unicode display name of the party.
CiscoUrlInfo	getUrlInfo()	Returns URL Info.
java.lang.String	getVoiceMailbox()	Returns voice mail box of the party.

Related Documentation

See Constant Field Values.

CiscoPickupGroup

CiscoPickupGroup is a new interface that represents a Pickup Group at the JTAPI layer. Currently, all a PickupGroup is a pair of String objects representing the Pickup Group's DN, and the Pickup Group's Partition.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.
8.0(1)	Following APIs are added: • getPickupGroupDN() • getPickupGroupPartition()

Declaration

public interface CiscoPickupGroup

Methods

Table 129: Methods in CiscoPickup Group

Interface	Method	Description
String	getPickupGroupDN()	Returns a String object that represents the number of the Pickup Group.
String	getPickupGroupPartition()	Returns a String object that represents the partition of the Pickup Group. It returns an empty String object if this pickup group does not belong to a partition.

Related Documentation

None

CiscoProvCallParkEv

CiscoProvCallParkEv event is delivered to providerobserver when a call is parked/unparked from any device in the cluster. To receive this event application should register using CiscoProvider.registerFeature() and CiscoProvFeatureID.MONITOR_CALLPARK_DN. User profile used by the application should have the Call Park Retrieval Allowed flag enabled to receive this event.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, CiscoProvFeatureEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Declaration

public interface CiscoProvCallParkEv extends CiscoProvFeatureEv

Fields

Table 130: Fields in CiscoProvCallParkEv

Interface	Field	Description
Static int	ID	—

Interface	Field	Description
Static int	PARK_STATE_ACTIVE	Indicates that a call is parked.
staticint	PARK_STATE_IDLE	Indicates that a call is unparked.
staticint	REASON_CALLPARK	Indicates that this event is due to call park.
staticint	REASON_CALLPARKREMAINDER	Deprecated This interface is deprecated due to a spelling error. Use the new interface REASON_CALLPARKREMINDER.
staticint	REASON_CALLPARKREMINDER	Indicates that the call is offered back to the parking party after call park reminder.
staticint	REASON_CALLUNPARK	Indicates that the call is unparked.

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 131: Methods in CiscoProvCallParkEv

Interface	Method	Description
int	getintCallIDValue()	Returns an integer representation of this object.
java.lang.String	getParkDN()	Returns where the call is parked.
java.lang.String	getParkedParty()	Returns the DN of the parked party.

Interface	Method	Description
java.lang.String	getParkedPartyPartition()	Returns the partition of the Parked Party.
java.lang.String	getParkingParty()	Returns the DN of the parking party.
java.lang.String	getParkingPartyPartition()	Returns the partition of the Parking party.
java.lang.String	getParkPartyPartition()	Returns the partition of park DN.
int	getReason()	Returns the reason of the event.
int	getState()	Returns the state of the call. Possible states are CiscoProvCallParkEv.PARK_STATE_IDLE CiscoProvCallParkEv.PARK_STATE_ACTIVE.

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoProvFeatureEv

getFeatureID

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoProvEv

The CiscoProvEv interface, which extends JTAPI's core javax.telephony.events.ProvEv interface, serves as the base interface for all Cisco-extended JTAPI Provider events. Every Provider-related event in this package extends this interface, directly or indirectly.

Interface History

Cisco Unified Communications Manager Release Number	Description
8.0(1)	Added new API getCiscoCause() which returns the CiscoCause for delivering the provider events.

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Subinterfaces

CiscoAddrActivatedEv, CiscoAddrActivatedOnTerminalEv, CiscoAddrAddedToTerminalEv, CiscoAddrCreatedEv, CiscoAddrRemovedEv, CiscoAddrRemovedFromTerminalEv, CiscoAddrRestrictedEv, CiscoAddrRestrictedOnTerminalEv, CiscoProvCallParkEv, CiscoProvConnToLeastPriorCtiServerEv, CiscoProvFallbackToPrimNwCompltdEv, CiscoProvFeatureEv, CiscoProvPrimNwReachableEv, CiscoProvTerminalCapabilityChangedEv, CiscoRestrictedEv, CiscoTermActivatedEv, CiscoTermCreatedEv, CiscoTermRemovedEv, CiscoTermRestrictedEv,

Declaration

public interface CiscoProvEv extends CiscoEv, javax.telephony.events.ProvEv

Fields

None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Interface	Method	Description
int	getCiscoCause ()	This method returns the cause to let application know why the event has been delivered.
Static final int	CiscoProvEv.CAUSE_NORMAL	This indicates the cause for non - EM login/logout scenarios. It will have an integer value of 0.
Static final int	CiscoProvEv.CAUSE_EM_LOGIN	This cause indicates an EM login on a terminal with a profile that is in the application's control list and/or with a user id that matches with the user id with which application has been started. It will have an integer value of 1.
Static final int	CiscoProvEv. CAUSE_EM_LOGOUT	This cause indicates an EM logout from a terminal with the profile that is in the application's control list and/or with a user id that matches with the user id with which application has been started. It will have an integer value of 2.
Static final int	CiscoProvEv. CAUSE_EM_LOGIN_PROFILE_ADD	This cause indicates a case where profile is added to the control list when it is already logged into a terminal. It will have an integer value of 3.
Static final int	CiscoProvEv. CAUSE_EM_LOGIN_PROFILE_REMOVE	This cause indicates a case where profile is removed from the control list when it is already logged into a terminal. It will have an integer value of 4.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

CiscoProvFeatureEv

The CiscoProvFeatureEv interface extends the com.cisco.jtapi.extensions.CiscoProvEv interface for provider events.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Subinterfaces

CiscoProvCallParkEv

Declaration

public interface CiscoProvFeatureEv extends CiscoProvEv

Fields

None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 132: Methods in CiscoProvFeatureEv

Interface	Method	Description
int	getFeatureID()	The feature ID for which the application wants to receive events.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See CiscoProvEv.

ProvEv.

CiscoProvFeatureID

This interface lists the features that registerFeature supports.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.
7.1(3)	Interface is enhanced to allow application register to get CiscoProvTerminalRegisteredEv and CiscoProvTerminalUnRegisteredEv events when terminal register and unregister respectively. CiscoProvTerminalRegisteredEv and CiscoProvTerminalUnRegisteredEv will be delivered to Provider observer when application registers for this feature

Declaration

public interface CiscoProvFeatureID

Fields

Table 133: Fields in CiscoProvFeatureID

Interface	Field	Description
static int	MONITOR_CALLPARK_DN	Used in the registerFeature interface in CiscoProvider to receive CiscoProvCallParkEv when a call gets parked or unparked from any device in the cluster.
public static final int	TERMINAL_REGISTER_UNREGISTER_EVENT_NOTIFY	Application can use to this to receive CiscoProvTerminalRegisteredEv and CiscoProvTerminalUnRegisteredEv

Sample Code

To register for Terminal Register and Unregister event notification:

```
try{JtapiPeer peer  =  JtapiPeerFactory.getJtapiPeer ( null );
} catch (JtapiPeerUnavailableException e){

}

MyProviderObserver providerObserver  =  new MyProviderObserver ();
Try{
  provider  =  peer.getProvider ( ipaddress;login = useid;passwd = password );
} catch (ProviderUnavailableException exp){
}
  if ( provider ! =  null ) {
    provider.addObserver ( providerObserver );
    provInService.waitTrue();
    System.out.Println("Enabling Register and Unregister events  ");
    try{
((CiscoProvider)provider).registerFeature(CiscoProvFeatureID.
TERMINAL_REGISTER_UNREGISTER_EVENT_NOTIFY);
   } catch (InvalidStateException ec){
   }

}
```

// CiscoProvTerminalRegisteredEv and CiscoProvTerminalUnRegisteredEv are delivered to Provider Observer.

```
class MyProviderObserver implements ProviderObserver {

  public synchronized void providerChangedEvent ( ProvEv [] eventList ) {
```

```
    try {
      if ( eventList ! =  null ) {
        for ( int i  =  0; i < eventList.length; i++ ) {
          if ( eventList[i] instanceof CiscoProvTerminalRegisteredEv ){
           CiscoProvTerminalRegisteredEv  ev  =  (CiscoProvTerminalRegisteredEv)

              eventList[i];
            System.out.Println( ev.getTerminal().getName() + " registered with
CUCM" );
          }
        }
     } catch (Exception eee){
}
```

Methods

None

Related Documentation

See Constant Field Values.

CiscoProvPickupCallAlertEv

CiscoProvPickupCallAlertEvent is a new interface being added with Call Pickup feature development. This event is fired whenever there is a call to be picked up in a pickup group that the provider is observing. See previous changes to CiscoProvider for information about how to register to observe a pickup group.

Cisco Unified Communications Manager Release Number	Description
8.0(1)	New interface.

Declaration

public interface CiscoProvPickupCallAlertEvent extends CiscoProvEv

Methods

Table 134: Methods of CiscoProvPickupCallAlertEv

Interface	Method	Description
String	getPickupGroupNumber()	This method returns the Pickup Group Number for which this event is being fired.
String	getPickupGroupPartition()	This method returns the Pickup Group Number for which this event is being fired.
CiscoCallID	getCallID()	This method returns the Call ID for the ringing call.

Interface	Method	Description
CiscoPartyInfo	getCallingPartyInfo()	This method returns a CiscoPartyInfo representing the calling party.CAVEAT: Currently, if the calling party is from out of cluster (External), it will still report as being Internal on the Address object inside of the CiscoPartyInfo.
CiscoPartyInfo	getCalledPartyInfo()	This method returns a CiscoPartyInfo representing the called party.

CiscoProvTerminalIPAddressChangedEv

This interface will be delivered to provider observers added by applications whenever the IP address of a terminal changes without the terminal getting unregistered.

Interface History

Cisco Unified Communications Manager Release Number	Description
9.0(1)	New interface.

Declaration

public interface CiscoProvTerminalIPAddressChangedEv extends CiscoProvEv

Fields

Table 135: Fields in CiscoProvTerminalIPAddressChangedEv

Interface	Field	Description
public Terminal	getTerminal()	Returns the Terminal that registered with Cisco Unified Communication Manager.
public int	getIPAddressingMode()	Returns the active IP Addressing mode of the terminal after the change. Based on this value, applications can query either the Ipv4 or the Ipv6 address of the terminal. Addressing mode may be any of the following constants: CiscoTerminal.IP_ADDRESSING_MODE_IPv4 CiscoTerminal.IP_ADDRESSING_MODE_IPv6 CiscoTerminal.IP_ADDRESSING_MODE_IPv4_v6

Interface	Field	Description
public InetAddress	getIPV4Address()	Returns the IPv4 address of the terminal. If the addressing mode is CiscoTerminal.IP_ADDRESSING_MODE_IPv6, this method will return null.
public InetAddress	getIPV6Address()	Returns the IPv6 address of the terminal. If the addressing mode is CiscoTerminal.IP_ADDRESSING_MODE_IPv4, this method will return null.

Methods

None

Related Documentation

None

CiscoProvTerminalMultiMediaCapabilityChangedEv

CiscoProvTerminalMultiMediaCapabilityChangedEv is a new interface that is notified to application as a Provider Event. when the video capability of the terminal changes, this interface is delivered to the provider observers added by applications.

Table 136: Interface History

Cisco Unified Communications Manager Release Number	Description
10.0(1)	New interface.

Declaration

public interface CiscoProvTerminalMultiMediaCapabilityChangedEv

com.cisco.jtapi.extensions.CiscoProvTerminalMultiMediaCapabilityChangedEv

Methods

Table 137: Methods in CiscoProvTerminalMultiMediaCapabilityChangedEv

Interface	Method	Description
Terminal	getTerminal()	This API returns the Terminal that is registered with Cisco UCM.

Interface	Method	Description
int	getVideoCapability()	This returns the video capability of the Terminal. The video capability can be: • CiscoMultiMediaCapabilityInfo.NONE • CiscoMultiMediaCapabilityInfo.VIDEO_ENABLED

CiscoProvTerminalRegisteredEv

This event is delivered to provider observer whenever a terminal registers with Cisco Unified Communication Manager. To receive this event, the application must use registerFeature API with CiscoFeatureID. TERMINAL_REGISTER_UNREGISTER_EVENT_NOTIFY. This event is delivered if a Terminal registers to Cisco Unified Communication Manager after the application registers for the feature using registerFeature API. During initialization time and JTAPI failover time the application can see this event for some the Terminals in the control list.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(3)	New interface.

Declaration

public interface CiscoProvTerminalRegisteredEv extends CiscoProvEv.

Fields

Table 138: Fields in CiscoProvTerminalRegisteredEv

Interface	Field	Description
Terminal	getTerminal()	Returns the terminal that registered with Cisco Unified Communications Manager.

Methods

None

Related Documentation

None

CiscoProvTerminalUnRegisteredEv

This event is delivered to provider observer when ever a terminal unregisters from Cisco Unified Communication Manager. To receive this event, the application must use the registerFeature API with CiscoFeatureID. TERMINAL_REGISTER_UNREGISTER_EVENT_NOTIFY.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(3)	New interface.

Declaration

public interface CiscoProvTerminalUnRegisteredEv extends CiscoProvEv.

Fields

Table 139: Fields in CiscoProvTerminalRegisteredEv

Interface	Field	Description
Terminal	getTerminal()	Returns the terminal that un-registered with Cisco Unified Communications Manager.
public final static int public final static int public final static int public final static int	• REASON_UNKNOWN • REASON_RESET • REASON_LOGIN • REASON_LOGOUT	• Indicates Terminal un-registered for unknown reason • Indicates Terminal un-registered due to rest • Indicates Terminal un-registered due to login • Indicates Terminal un-registered due to logout
Int	getReason()	Returns the reason of un-register. The return value is one of the above defined reasons.

Methods

None

Related Documentation

None

CiscoProvider

The CiscoProvider interface extends the Provider interface with additional Cisco capabilities.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.
8.0(1)	Enhanced to have the following: • New API registerPickupAlert(String pickupDn, String pickupPartition) • unregisterPickupAlert(String pickupDn, String pickupPartition) which allow the application to register and unregister for the reception of Call Pickup events. • CiscoProvPickupCallAlertEvent, which is a provider event what the application receives when they register for events using the previously mentioned API • ProviderCallPickupRegistrationClosedEv, which is a provider event used to alert the application if something happens that would close the registration event, such as the pickup group being removed from the Unified CM admin panel.
10.0(1)	A new method is added: getClusterID() this returns the clusterID enterprise parameter configured for the cluster as a string.
14SU3	Enhanced to have the following: • New APIs setLeastPriorityCtiServer(String leastPriorityCtiServer) • setLeastPriorityCtiServer(String leastPriorityCtiServer, int fallbackInitiationTime) • getLeastPriorityCtiServer() • isCtiServerAvailable(String server) • initiateFallback() • initiateFallback(String server)

Superinterfaces

CiscoObjectContainer, javax.telephony.Provider.

Declaration

public interface CiscoProvider extends javax.telephony.Provider, CiscoObjectContainer

Fields

None

Inherited Fields

From Interface javax.telephony.Provider

IN_SERVICE, OUT_OF_SERVICE, SHUTDOWN

New Error Codes

CTIERR_ALREADY_REGISTERED

This error code indicates that the Pickup Group attempting to be registerred for has already been registerred by this provider.

CTIERR_REGISTRATION_NOT_FOUND

This error code indicates that an unregister attempt failed because the Pickup Group specified was not registerred for previously.

CTIERR_INVALID_PICKUPGROUP

This error code indicates that the Pickup Group specified in the register or unregister event is not valid.

Methods

Table 140: Methods in CiscoProvider

Interface	Method	Description
CiscoTerminal	createTerminal (java.lang.Stringname)	Returns an instance of the CiscoTerminal class which corresponds to the given name. Application must have sufficient capability otherwise PrivilegeViolationException gets thrown CiscoProvider.createTerminal(). **Pre-conditions** this.getState() = = Provider.IN_SERVICE **Post-conditions** Create CiscoTerminal corresponding to name; terminal is an element of this.getTerminals(). **Parameters** • name—The name of desired CiscoTerminal object. **Throws** javax.telephony.InvalidArgumentException—The name provided does not correspond to a name of any CiscoMediaTerminal known to the Provider or within the Provider's domain. javax.telephony.InvalidStateException—The provider is not inService. PreviledgeVoilationException—The provider does not have sufficient capbilitly i.e. CiscoProviderCapabilities.canObserveAnyTerminal() returns false call.getState() = = Call.INVALID

Interface	Method	Description
void	deleteTerminal (CiscoTerminalterminal)	Removes the CiscoTerminal Object from providers control. Application must have created this terminal using Provider.createTerminal() interface otherwise PreviledgeVoilationException gets thrown. CiscoProvider.deleteTerminal(). **Pre-conditions** this.getState() = = Provider.IN_SERVICE **Post-conditions** CiscoTerminal Object deleted from providers list of terminal. Terminal is not element of this.getTerminals() any more and Addresses belonging to terminal get deleted. **Parameters** • terminal—The referece to the desired CiscoTerminal object to be deleted. **Throws** javax.telephony.InvalidArgumentException—The terminal provided is not element of this.getTerminals() or terminal is not provider domain. PrivilegeViolationException—The terminal given in the argument is not a terminal created using Provider.createTerminal() method. Applications can delete only those terminal which are created using Provider.createTerminal() interface.
javax.telephony.Address	getAddress (java.lang.Stringnumber, java.lang.Stringpartition)	Returns an address object corresponding to the number and partition that is passed in the method. The address object will be unique for a particular number, partition combination. **Throws** javax.telephony.InvalidArgumentException
int	getAppDSCPValue ()	Gets the DSCP value from the provider by using CiscoProvider.getAppDSCPValue(). **Pre-conditions** this.getState() = = Provider.IN_SERVICE **Post-conditions** The method will return the integer value of the DSCP value for applications set by CTI.
CiscoCall	getCall (CiscoRTPHandlertpHandle)	Returns call object with the RTPHandle associated with a specific terminal.

Interface	Method	Description
CiscoCall	getCall (intcallleg)	Returns CiscoCall present in provider domain and the call object with the RTPHandle associated with a specific terminal. This method may return null if this RTPHandle is no longer associated with any call or if there was no callObserver added on the terminal at the time when CiscoCallOpenLogicalChannelEv which contained this handle is sent to applications. **Throws** javax.telephony.InvalidStateException
boolean	getCallbackGuardEnabled ()	None
boolean	isFIPSCompliantJTAPI ()	Returns true if JTAPI is running in FIPS Compliance mode. This means that the application has explicitly requested FIPS compliance, and that the libraries are running properly.
boolean	isFIPSCompliantCUCM ()	Returns true if the Unified CM server is running in FIPS Compliance mode.
CiscoIntercomAddress[]	getIntercomAddresses ()	Returns array of CiscoInterComAddress present in provider domain.

Interface	Method	Description
CiscoMediaTerminal	getMediaTerminal (java.lang.Stringname)	Returns an instance of the CiscoMediaTerminal class which corresponds to the given name. Each CiscoMediaTerminal has a unique name associated with it, which is assigned to it by the JTAPI implementation. If no CiscoMediaTerminal is available for the given name within the Provider domain, this method throws the InvalidArgumentException. This CiscoMediaTerminal is contained in the arrays generated by Provider.getTerminals() and CiscoProvider.getMediaTerminals(). **Pre-conditions** Let CiscoMediaTerminal terminal = this.getMediaTerminal(name); terminal is an element of this.getTerminals(); terminal is an element of this.getMediaTerminals(); **Post-conditions** Let CiscoMediaTerminal terminal = this.getMediaTerminal(name); terminal is an element of this.getTerminals(); terminal is an element of this.getMediaTerminals(); **Parameters** • name—The name of desired CiscoMediaTerminal object. **Throws** javax.telephony.InvalidArgumentException—The name provided does not correspond to a name of any CiscoMediaTerminal known to the Provider or within the Provider domain.

Interface	Method	Description
CiscoMediaTerminal[]	getMediaTerminals ()	Returns an array of CiscoMediaTerminals associated with the Provider and within the Provider local domain. Each CiscoMediaTerminal possesses a unique name, which is assigned to it by the JTAPI implementation. If there are no CiscoMediaTerminals associated with this Provider, then this method returns null. This array is a subset of the array returned by Provider.getTerminals(). **Post-conditions** Let CiscoMediaTerminal[] terminals = this.getMediaTerminals() terminals = = null or terminals.length > = 1 if terminals ! = null, terminals is a subset of this.getTerminals () **Throws** javax.telephony.ResourceUnavailableException—Indicates the number of media terminals present in the Provider is too great to return as a static array.
CiscoPickupGroup[]	getRegisteredPickupGroups ()	This method returns an array of CiscoPickupGroup objects that represents all of the Pickup Groups that this provider is currently registerred to observe. **Parameters** A String object that represents the number of the Pickup Group to be registerred for, and another String object that represents the partition that the Call Pickup Group is in.
java.lang.String	getVersion ()	None
void	registerFeature (intfeatureID)	Registers a particular feature for which application gets Provider events. Applications should pass in the featureID of the softkey. Current supported features are listed in CiscoProvFeatureID interface. **Throws** javax.telephony.InvalidStateException javax.telephony.PrivilegeViolationException javax.telephony.InvalidArgumentException

Interface	Method	Description
void	registerPickupAlert (String pickupDN, String pickupPartition)	This method tells the Provider to register for receiving Call Pickup events. After this method is called, Call Pickup events for the specified Call Pickup Group will be sent to all JTAPI observers under this provider. **Parameters** • A String object that represents the number of the Pickup Group to be registerred for, and another String object that represents the partition that the Call Pickup Group is in. • The pickupPartition can be passed in as an empty String (“”) or null if the pickup group does is not in any partition. • An application can use the new CiscoPickupGroup object in place of the pair of Strings for either method.
void	registerPickupAlert (CiscoPickupGroup pickupGroup)	This method tells the Provider to register for receiving Call Pickup events. After this is called, Call Pickup events for the specified Call Pickup Group will be sent to all JTAPI observers under this provider. **Parameters** • A String object that represents the number of the Pickup Group to be registerred for, and another String object that represents the partition that the Call Pickup Group is in. • The pickupPartition can be passed in as an empty String (“”) or null if the pickup group does is not in any partition. • An application can use the new CiscoPickupGroup object in place of the pair of Strings for either method.

Interface	Method	Description

Void	setCallbackGuardEnabled (booleanenabled)	Enables or disables try/catch logic for observer callbacks. In order to protect itself from application exceptions in observer callbacks, the Provider normally guards all invocations of application interfaces (e.g. observers) with the following code:
		`try {observer.callStateChanged ( ... );`
		`} catch ( Throwable t ) {`
		`// log the exception here`
		`}`
		This isolates application errors from the JTAPI implementation, allowing easier troubleshooting, since the JTAPI implementation can note the unhandled exception and continue operating.
		Some errors are considered non-recoverable and will be re-thrown by JTAPI, generally resulting in application exit. Such errors include ThreadDeath, OutOfMemoryError, and StackOverflowError.
		Applications wishing to trap errors within JTAPI threads should create a subclass of ThreadGroup and initialize JTAPI from a thread within that ThreadGroup.
		By overriding the ThreadGroup.uncaughtException () method, the application can be made aware of all unrecoverable errors thrown on JTAPI threads. In some cases, JTAPI's aggressive error-catching approach may make it more difficult to troubleshoot applications within a java debugger.
		Microsoft Visual J++ version 6.0, for example, does not handle breakpoints within application observer callbacks properly if JTAPI catches Throwable. In such cases, JTAPI application developers may choose to disable the internal JTAPI try/catch logic.
		Note Disabling callback guards in this manner is only intended for use while troubleshooting applications, and never for use in production environments. By default, callback guards are always enabled.
		Parameters
		• enabled—if true, callback guard will be enabled; if false, callback guard will be disabled.

Void	unregisterFeature (intfeatureID)	Unregisters a particular feature.

Interface	Method	Description
Void	unregisterPickupAlert (String pickupDN, String pickupPartition)	This method will tell the Provider to unregister for receiving Call Pickup events. After this is called, Call Pickup events for the specified Call Pickup Group will no longer be sent to all JTAPI observers under this provider. **Parameters** • A String object that represents the number of the Pickup Group to be registerred for, and another String object that represents the partition that the Call Pickup Group is in. • The pickupPartition can be passed in as an empty String (“”) or null if the pickup group does is not in any partition. • An application can use the new CiscoPickupGroup object in place of the pair of Strings for either method.
Void	unregisterPickupAlert (CiscoPickupGroup pickupGroup)	This method will tell the Provider to unregister for receiving Call Pickup events. After this is called, Call Pickup events for the specified Call Pickup Group will no longer be sent to all JTAPI observers under this provider. **Parameters** • A String object that represents the number of the Pickup Group to be registerred for, and another String object that represents the partition that the Call Pickup Group is in. • The pickupPartition can be passed in as an empty String (“”) or null if the pickup group does is not in any partition. • An application can use the new CiscoPickupGroup object in place of the pair of Strings for either method.
CiscoRemoteTerminal[]	getRemoteTerminals ()	This method returns an array of CiscoRemoteTerminal associated with the Provider and within the Provider's domain. Each CiscoRemoteTerminal possesses an unique name, which is assigned to it by the JTAPI implementation. If there is no CiscoRemoteTerminals associated with this Provider, this API will return null. This array is a subset of the array returned by Provider.getTerminals().

Interface	Method	Description
CiscoRemoteTerminal	getRemoteTerminal (String name)	This method returns an instance of the CiscoRemoteTerminal class which corresponds to the given name. Each CiscoRemoteTerminal has an unique name associated to it, which is assigned by the JTAPI implementation. If no CiscoRemoteTerminal is available for the given name within the Provider's domain, this API throws the InvalidArgumentException. This CiscoRemoteTerminal is contained in the arrays generated by Provider.getTerminals() and CiscoProvider.getRemoteTerminals().
String	getClusterID ()	This method returns the clusterID enterprise parameter configured for the cluster as a string. If this enterprise parameter is changed CTIManager service and other Cisco Communication Manager services need to be restarted. **Pre-conditions** Provider.getState() = = Provider.IN_SERVICE If pre-condition is not met, InvalidStateException is thrown. Default value is StandAloneCluster.
Void	setLeastPriorityCtiServer(String leastPriorityCtiServer)	This method allows application to mark a CTI server as least priority. Least Priority indicates, JTAPI would connect to the CTI server only in the case when no other CTI Server configured by application is reachable. JTAPI would also initiate a forceful fallback once one of the CTI servers are reachable again (600 seconds post this event). Basically, all other CTI servers configured and not marked as least priority have equal weightage. Application can only configure one CTI server as least priority.

Interface	Method	Description
void	setLeastPriorityCtiServer(String leastPriorityCtiServer, int fallbackInitiationTime)	This API allows application to mark a CTI server as least priority. Least Priority indicates, JTAPI would connect to the CTI server only in the case when no other CTI Server configured by application is reachable. JTAPI would also initiate a forceful fallback once one of the CTI servers arereachable again. Basically, all other CTI servers configured and not marked as least priority have equal weightage. Application can only configure one CTI server as least priority. This method is overloaded to take additional parameter to specify time (in seconds) after which a forceful fallback is to be initiated once primary network becomes reachable. Fallback initiation time is defined as below: • Default value : 300 seconds • Min value : 120 seconds • Max value: 600 seconds • Default value would be taken if specified value is out of the range.
String	getLeastPriorityCtiServer()	This API returns the least priority CTI server as configured by application by invoking <CODE>CiscoProvider.setLeastPriorityCtiServer((server)</CODE>.
boolean	isCtiServerAvailable(String server)	This API allows application to query if one of the configured CTI servers is reachable.
void	initiateFallback()	This API allows application to invoke a fallback when connected to CTI server which was previously identified as least priority by invoking <CODE>CiscoProvider.setLeastPriorityCtiServer(server)</CODE>. Application can invoke this in case one of the primary network CTI Server becomes available and application is ready to do a fallback before the configured/default fallback timer expires.
void	initiateFallback(String server)	This API allows application to invoke a fallback when connected to CTI server which was previously identifie as least priority by invoking <CODE>CiscoProvider.setLeastPriorityCtiServer(server)</CODE>. Application can invoke this in case one of the primary network CTI Server becomes available and application is ready to do a fallback before the configured/default fallback timer expires. This method is overloaded to take additional parameter to specify the server to which application needs to fallback to.

Inherited Methods

From Interface javax.telephony.Provider

addObserver, createCall, getAddress, getAddressCapabilities, getAddressCapabilities, getAddresses, getCallCapabilities, getCallCapabilities, getCalls, getCapabilities, getConnectionCapabilities, getConnectionCapabilities, getName, getObservers, getProviderCapabilities, getProviderCapabilities, getState, getTerminal, getTerminalCapabilities, getTerminalCapabilities, getTerminalConnectionCapabilities, getTerminalConnectionCapabilities, getTerminals, removeObserver, shutdown

From Interface com.cisco.jtapi.extensions.CiscoObjectContainer

getObject, setObject

Related Documentation

None

CiscoProviderCapabilities

This interface defines the Cisco-specific provider capabilities that Cisco Unified JTAPI offers.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Added support for the method canSupportIPv6()T.
8.0(1)	Enhanced by adding new API canAutoPickup(), which lets the application determine whether or not the CUCM service parameter "Auto Call Pickup Enabled" is set to true or false. This service parameter has an impact on the events and behavior of Call Pickup, and applications can use this new API to determine if it's enabled or not, and act accordingly.

Superinterfaces

javax.telephony.capabilities.ProviderCapabilities

Declaration

public interface CiscoProviderCapabilities extends javax.telephony.capabilities.ProviderCapabilities

Methods

Table 141: Methods in CiscoProviderCapabilities

Interface	Method	Description
boolean	canAutoPickup()	This method returns a boolean value representing whether or CUCM service parameter "Auto Call Pickup Enabled" is set ot true or false.
boolean	canObserveAnyTerminal()	This method checks whether the user has been provisioned in the Cisco Unified Communications Manager with the privilege to observe any Terminal (and its addresses) in the system. Such Terminals and Addresses do not get returned as part of the list that JTAPI initializes at startup. The provider obtained with the login for a user with such privileges can be determined from the canObserverAnyTerminal method call in ProviderCapabilities. Returns True if the user can observe any Terminal in the system, or false if the user can only observe Terminals and Addresses in the control list.

	Example
	`Provider p = peer.getProvider( loginString );`
	`ProviderCapabilities caps = p.getCapabilities ();`
	`if ( caps instanceof CiscoProviderCapabilities )`
	`{`
	`boolean canObserveAnyTerminal =`
	`((CiscoProviderCapabilities)caps).canObserveAnyTerminal ();`
	`boolean canMonitorParkDN =`
	`((CiscoProviderCapabilities)caps).canMonitorParkDNs ();`
	`boolean canModifyCallingPN =`
	`((CiscoProviderCapabilities)caps).canModifyCallingParty ();`
	`boolean canRecordCalls =`
	`((CiscoProviderCapabilities)caps).canRecord();`
	`boolean canMonitorCalls =`
	`((CiscoProviderCapabilities)caps).canMonitor();`
	`}`	

boolean	canMonitorParkDNs()	This method checks whether the user has been provisioned in the Cisco Unified Communications Manager to monitor park DNs. Returns True if the user can monitor park DNs, or false otherwise.
boolean	canModifyCallingParty()	This method checks whether the user has been provisioned in the Cisco Unified Communications Manager to modify the calling party number of a call. Returns True if the user can modify the calling party number, or false otherwise.

Interface	Method	Description
boolean	canRecord()	This method checks whether the user has been provisioned in the Cisco Unified Communications Manager to record calls. Only users in 'Standard CTI Allow Call Recording' user group can record calls. Returns True if the user belongs to the group.
boolean	canMonitor()	This method checks whether a user has been provisioned in the Cisco Unified Communications Manager to monitor calls. Only users in 'Standard CTI Allow Call Monitoring' user group can initiate call monitoring request. Returns True if the user belongs to the group.
boolean	canSupportIPv6()	This interface returns true if Enterprise Parameter “Enable IPv6” is enabled and false otherwise.

Inherited Methods

From Interface javax.telephony.capabilities.ProviderCapabilities

isObservable

Related Documentation

See canObserveAnyTerminal().

CiscoProviderCapabilityChangedEv

Application provider observers receive this event when a user gets added or removed from user groups (capabilitied) in Cisco Unified Communications Manager. The methods for this event let you check which capabilities changed.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Added hasIPv6CapabilityChanged() method.

Declaration

public interface CiscoProviderCapabilityChangedEv

Fields

Table 142: Fields in CiscoProviderCapabilityChangedEv

Interface	Field	Description
staticint	ID	None
staticint	MODIFY_CGPN	Deprecated This constant is not returned by any interface, should not be used by application.
staticint	MONITOR_PARKDN	Deprecated This constant is not returned by any interface, should not be used by application.
staticint	SUPERPROVIDER	Deprecated This constant is not returned by any interface, should not be used by application.

Methods

Table 143: Methods in CiscoProviderCapabilityChangedEv

Interface	Method	Description
CiscoProviderCapabilities	getCapability()	This method returns the current CiscoProviderCapabilities object for the user.
boolean	hasIPv6CapabilityChanged()	This method can be used by applications to determine whether Enable IPv6 Enterprise Parameter has changed. **Pre-conditions** this.getState() = = Provider.IN_SERVICE **Post-conditions** The method returns True when the Enable IPv6 Enterprise parameter gets changed; otherwise it returns False.
boolean	hasModifyCallingPartyChanged()	This method checks whether the “modify Calling Party" privilege has changed. **Pre-conditions** provider.getState() = = Provider.IN_SERVICE

Interface	Method	Description
boolean	hasMonitorCapabilityChanged()	This method checks whether the monitor capability of a user has changed. **Pre-conditions** provider.getState() = = Provider.IN_SERVICE
boolean	hasMonitorParkDNChanged()	This method checks whether the "monitor Park DN" privilege has changed. **Pre-conditions** provider.getState() = = Provider.IN_SERVICE
boolean	hasObserveAnyTerminalChanged()	This method checks whether the "can control any terminal" privilege has changed **Pre-conditions** provider.getState() = = Provider.IN_SERVICE
boolean	hasRecordingCapabilityChanged()	This method checks whether the recording capability of the has changed. **Pre-conditions** provider.getState() = = Provider.IN_SERVICE

Related Documentation

See Constant Field Values.

CiscoProviderObserver

Implement this interface to receive CiscoProvEv events such as CiscoAddrCreatedEv and CiscoTermCreatedEv when observing a Provider via the Provider.addObserver method.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.ProviderObserver

Declaration

public interface CiscoProviderObserver extends javax.telephony.ProviderObserver

Methods

None

Inherited Methods

From Interface javax.telephony.ProviderObserver

providerChangedEvent

Related Documentation

See CiscoAddrCreatedEv and CiscoTermCreatedEv.

CiscoProvTerminalCapabilityChangedEv

This event is delivered to the Provider when Terminal Capability is changed. This event is provided on application observer .

Interface History

Cisco Unified Communications Manager Release Number	Description
7.0(1)	Added event.
7.0(1)	Modified the CiscoTerminal[] interface so that only CiscoMediaTerminals or CiscoRouteTerminals gets returned.

Superinterfaces

CiscoEv, CiscoProvEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Declaration

public interface CiscoProvTerminalCapabilityChangedEv extends CiscoProvEv

Fields

Table 144: Fields in CiscoProvTerminalCapabilityChangedEv

Interface	Field	Description
staticint	ID	None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 145: Methods in CiscoProvTerminalCapabilityChangedEv

Interface	Method	Description
CiscoTerminal[]	getTerminals()	Returns an array of CiscoTerminals whose capabilities have changed. In Cisco Unified Communications Manager Release 7.0(1), CiscoTerminal[] interface was modified so that only CiscoMediaTerminals or CiscoRouteTerminals get returned.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

None

CiscoProvTerminalRemoteDestinationChangedEv

CiscoProvTerminalRemoteDestinationChangedEv is a new interface exposed to the applications as a Provider Event. JTAPI sends this event to the application's provider observer when any of Remote Destination information is changed on a CiscoRemoteTerminal in the provider domain. This event contains the latest list of Remote Destinations at a given time. And depending on the request or operation done on these remote destinations, single or multiple CiscoProvTerminalRemoteDestinationChangedEv event(s) can be generated from the change notification(s).

Methods

Interface	Method	Description
CiscoRemoteTerminal	getTerminal()	This method returns the CiscoRemoteTerminal object for which its remote destination has changed.
CiscoRemoteDestinationInfo[]	getRemoteDestinations()	This method returns an array of CiscoRemoteDestinationInfo objects representing the latest/current list of remote destinations associated to the CiscoRemoteTerminal at the time when this change notification event took place.
boolean	isMyAppLastToSetActiveRD()	This method can be used by application to determine whether it is the last application to set active remote destination for the CiscoRemoteTerminal.

CiscoRecorderInfo

This interface provides information about the recorder in a recording session. When a recording session is active, this interface gives information about the recording device.

Interface History

Cisco Unified Communications Manager Release Number	Description
12.5(1)	A new method getMultiForkingRecorderInfo() is added which returns an array (maximum size 5) of CiscoMultiForkingRecorderInfo[]. The following are the four new methods inside getMultiForkingRecorderInfo() that provide the information about each recorder. • getRecorderURI() • getRecorderErrorMsg() • getRecorderType() • getRecorderStatus()
10.0(1)	Four new methods are added: • getMediaForkingDeviceType() • getMediaForkingDeviceName() • getProtocolReferenceGUID() • getMediaForkingClusterID()
9.0(1)	A new method, getRecordingType(), is added.
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoRecorderInfo

Fields

None

Methods

Table 146: Methods in CiscoRecorderInfo

Interface	Method	Description
CiscoAddress	getAddress()	Returns the recorder address.
public int	getRecordingType()	This method returns the recording type that was used to start the recording.
java.lang.String	getTerminalName()	Returns the terminal name of the recording device.

Interface	Method	Description
public int	getMediaForkingDeviceType()	This interface returns the Media Forking Device Type for Gateway Recording. The forking device type can be: CiscoCall.CALL_RECORDING_MEDIA_FORKING _DEVICE_TYPE_NONE = 0 CiscoCall.CALL_RECORDING_MEDIA_FORKING _DEVICE_TYPE_PHONE = 1 CiscoCall.CALL_RECORDING_MEDIA_FORKING _DEVICE_TYPE_GW = 2
java.lang.String	getMediaForkingDeviceName()	This interface returns the Forking Device Name for Gateway Recording.
java.lang.String	getProtocolReferenceGUID()	This interface returns the Protocol Call Reference GUID for Gateway Recording.
java.lang.String	getMediaForkingClusterID()	This interface returns the Forking Cluster ID for Gateway Recording.

Range of Values

The range of values returned by the getRecordingType() method is defined on the CiscoCall object:

CiscoCall.CALL_RECORDING_TYPE_NONE

CiscoCall.CALL_RECORDING_TYPE_APPLICATION_INITIATED_SILENT

CiscoCall.CALL_RECORDING_TYPE_AUTOMATIC

CiscoCall.CALL_RECORDING_TYPE_USER_INITIATED_FROM_APPLICATION

CiscoCall.CALL_RECORDING_TYPE_USER_INITIATED_FROM

Related Documentation

None

CiscoRemoteDestinationInfo

CiscoRemoteDestinationInfo is a new interface that contains the information about a remote destination on a CiscoRemoteTerminal. Applications can get a list of all associated remote destinations from the return array of CiscoRemoteTerminal.getAllRemoteDestinations().

Methods

Interface	Method	Description
String	getRemoteDestinationName()	This method returns the remote destination name.
String	getRemoteDestinationNumber()	This method returns the remote destination number.

Interface	Method	Description
boolean	getIsActiveRD()	This method returns whether the remote destination is an active remote destination or not.

CiscoRemoteTerminal

CiscoRemoteTerminal is a new interface that extends the interface of CiscoTerminal. A CiscoRemoteTerminal is a special kind of CiscoTerminal representing the CTI Remote Device and Jabber/CUCSF (Cisco Unified Client Services Framework) Device in extend mode. It allows applications to monitor remote destinated devices such as PSTN, PBSX, and Mobiles phones.

Interface History

Cisco Unified Communications Manager Release Number	Description
9.0(1)	A new interface, CiscoRemoteTerminal, is added.

Declaration

public interface CiscoRemoteTerminal

Methods

Interface	Method	Description
CiscoRemoteDestinationInfo[]	getAllRemoteDestinations ()	This method will return an array of CiscoRemoteDestinationInfo representing all remote destinations of the CiscoRemoteTerminal, or null if none. Note that CiscoProvider must be in IN_SERVICE state, otherwise InvalidStateException will be thrown.
CiscoRemoteDestinationInfo[]	getActiveRemoteDestinations ()	This method will return an array CiscoRemoteDestinationInfo representing all active remote destinations of the CiscoRemoteTerminal, or null if none. Note that CiscoProvider must be in IN_SERVICE state, otherwise InvalidStateException will be thrown.
void	setActiveRemoteDestination (String remoteDestinationNumber, boolean isActiveRD)	This method will set/unset an active remote destination of the CiscoRemoteTerminal based on the remote destination number. Note that CiscoProvider must be in IN_SERVICE state, otherwise InvalidStateException will be thrown. Also note that the Remote Destination Number must be of a valid associated remote destination, and if the remoteDestinationNumber parameter is null, it will throw InvalidArgumentException.

Interface	Method	Description
void	addRemoteDestination (String remoteDestinationName, String remoteDestinationNumber, boolean isActiveRD)	This method will add a new remote destination to the CiscoRemoteTerminal. Note that CiscoProvider must be in IN_SERVICE state, otherwise InvalidStateException will be thrown. And if either the remoteDestinationNumber or remoteDestinationName parameter is null, it will throw InvalidArgumentException.
void	removeRemoteDestination (String remoteDestinationNumber)	This method will remove a remote destination from the CiscoRemoteTerminal based on the remote destination number. Note that CiscoProvider must be in IN_SERVICE state, otherwise InvalidStateException will be thrown. Also note that the Remote Destination Number must be of a valid associated remote destination, and if the remoteDestinationNumber parameter is null, it will throw InvalidArgumentException.
void	removeAllRemoteDestinations ()	This API will remove all associated remote destinations from the CiscoRemoteTerminal. Note that CiscoProvider must be in IN_SERVICE state, otherwise InvalidStateException will be thrown.
void	updateRemoteDestinationName (String remoteDestinationNumber, String remoteDestinationName)	This method will update the name of a remote destination of the CiscoRemoteTerminal based on the remote destination number. Note that CiscoProvider must be in IN_SERVICE state, otherwise InvalidStateException will be thrown. Also note that the Remote Destination Number must be of a valid associated remote destination, and if the remoteDestinationNumber parameter is null, it will throw InvalidArgumentException.
void	updateRemoteDestinationNumber (String remoteDestinationNumber, StringnewRemoteDestinationNumber)	This method will update the number of a remote destination of the CiscoRemoteTerminal based on the remote destination number. Note that CiscoProvider must be in IN_SERVICE state, otherwise InvalidStateException will be thrown. Also note that the Remote Destination Number must be of a valid associated remote destination, and if either the remoteDestinationNumber or newRemoteDestinationNumber parameter is null, it will throw InvalidArgumentException.

Interface	Method	Description
void	updateRemoteDestination (String remoteDestinationNumber, String remoteDestinationName, String newRemoteDestinationNumber, boolean isActiveRD)	This method will update a remote destination of the CiscoRemoteTerminal based on the remote destination number. It can update any or all of its remote destination name, remote destination number, and isActiveRD at the same time. Note that CiscoProvider must be in IN_SERVICE state, otherwise InvalidStateException will be thrown. Also note that the Remote Destination Number must be of a valid associated remote destination, and if the remoteDestinationNumber parameter is null, it will throw InvalidArgumentException.
boolean	isRegisteredByThisApp ()	This method will return true if this application issued a successful registration request to register this terminal's device in Extend mode; return false otherwise. It will get set to true until this application unregisters the device.
int	getRegistrationType ()	This method will return the registration type with which this terminal's device has been registered in. The registration type returned can be one of the following: • CiscoRemoteTerminal. EXTEND_MEDIA_REGISTRATION • CiscoRemoteTerminal. NO_EXTEND_MEDIA_REGISTRATION
boolean	isMyAppLastToSetActiveRD ()	This method will return true if this application is the last application to set active remote destination for the CiscoRemoteTerminal; return false otherwise. Note that CiscoProvider must be in IN_SERVICE state, otherwise InvalidStateException will be thrown.

Parameters

String remoteDestinationName

The name of the specified remote destination.

String remoteDestinationNumber

The number of the specified remote destination.

boolean isActiveRD

The flag to indicate whether the specified remote destination is an active remote destination or not.

Data Type

public static final int

EXTEND_MEDIA_REGISTRATION = 8

This registration type applies to CUCSF device that is registered in Extend mode, which as a result is representing as a CiscoRemoteTerminal.

NO_EXTEND_MEDIA_REGISTRATION = 0

This registration type applies to non-CUCSF device that is static registered and is representing as CiscoRemoteTerminal, such as CTI Remote Device.

New Error Codes

CiscoJtapiException.CTIERR_INVALID_REMOTE_DESTINATION_NUMBER (0x8CCC0121)

CiscoJtapiException.CTIERR_DUPLICATE_REMOTE_DESTINATION_NUMBER (0x8CCC0122)

CiscoJtapiException.CTIERR_REMOTEDESTINATION_LIMIT_EXCEEDED (0x8CCC0123)

CiscoJtapiException. CTIERR_REMOTE_DEVICE_REQUEST_FAILED_ACTIVE_RD_NOT_SET (0x 8CCC0124)

CiscoJtapiException.CTIERR_ENDUSER_NOT_ASSOCIATED_WITH_DEVICE (0x 8CCC0126)

CiscoJtapiException.CTIERR_DEVICE_ALREADY_REGISTERED_NONEXTEND (0x 8CCC0127)

CiscoJtapiException.CTIERR_MEDIA_ALREADY_TERMINATED_EXTEND (0x 8CCC0128)

CiscoJtapiException.CTIERR_INVALID_REMOTE_DESTINATION_NAME (0x8CCC0130)

Sample Code

```
Sample Code:
public static void main(String[] args) {
  try
  {
    JtapiPeer peer  =  JtapiPeerFactory.getJtapiPeer(null);
    String provStr  =  provName + ";login = " + login + ";passwd = " + passwd;
    Provider myProvider  =  peer.getProvider(provStr);
    MyProviderObserver providerObserver  =  new MyProviderObserver();
    String myDevName  =  "CTIRD-A";
    String temp  =  “”;
    if (myProvider ! =  null)
    {
      myProvider.addObserver(providerObserver);
      provInService.waitTrue();
      CiscoRemoteTerminal rTerm  =
          (CiscoRemoteTerminal)(myProvider.getTerminal(myDevName));
      if (rTerm ! =  null)
      {
        CiscoRemoteDestinationInfo[] remoteDestinations  =
            rTerm.getAllRemoteDestinations();
        CiscoRemoteDestinationInfo[] activeRemoteDestinations  =
            rTerm.getActiveRemoteDestinations();
```

```
        System.out.println("CTI Remote Device Name: " + rTerm.getName());
        if (remoteDestinations ! =  null && remoteDestinations.length ! =  0)
        {
          System.out.println("Number of associated Remote Destinations (RD): " +
            remoteDestinations.length);
          for (int i  =  0; i < remoteDestinations.length; i++)
          {
            System.out.println("RD["+i+"] Name: " +
              remoteDestinations[i].getRemoteDestinationName());
            System.out.println("RD["+i+"] Number: " +
              remoteDestinations[i].getRemoteDestinationNumber());
            System.out.println("RD["+i+"] IsActiveRD: " +
              remoteDestinations[i].getIsActiveRD());
            temp  =  remoteDestinations[i].getRemoteDestinationName();
            if (temp ! =  null && temp.equalsIgnoreCase("MyOffice"))
            {
              temp  =  remoteDestinations[i].getRemoteDestinationNumber();
              rTerm.updateRemoteDestinationNumber(temp,  "9498231202");
              rTerm.setActiveRemoteDestination("9498231202",  true);
            }
          }
          rTerm.addRemoteDestination("MyHome",  "6267978244",  false);
        }
        else
        {
          System.out.println("There is no associated Remote Destinations (RD)");
        }
      }
      else
      {
        System.out.println("There is no CTI Remote Device of " + myDevName +
          " in this provider");
      }
    }
    else
    {
      System.out.println("Cannot create provider");
    }
  }
  catch (Exception e)
  {
    System.out.println("Exception caught for getting CTI Remote Device RD info!
" + e);
    if (e instanceof PlatformException)
    {
      switch (((CiscoJtapiException) e).getErrorCode())
      {
      case CiscoJtapiException.CTIERR_INVALID_REMOTE_DESTINATION_NUMBER:
        System.out.println("Invalid RD Number");
        break;
      case CiscoJtapiException. CTIERR_DUPLICATE_REMOTE_DESTINATION_NUMBER:
        System.out.println("Duplicated RD Number");
        break;
      }
    }
  }
}
...
```

CiscoRestrictedEv

The CiscoRestrictedEv event is the parent class for the CiscoAddrRestrictedEv and CiscoAddrRestrictedOnTerminalEv events. This is the base class for restricted events and defines the cause codes for all restricted events.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Subinterfaces

CiscoAddrRestrictedEv, CiscoAddrRestrictedOnTerminalEv

Declaration

public interface CiscoRestrictedEv extends CiscoProvEv

Fields

Table 147: Fields in CiscoRestrictedEv

Interface	Field	Description
staticint	CAUSE_UNKNOWN	The Terminal is restricted for an unknown reason.
staticint	CAUSE_UNSUPPORTED_DEVICE_CONFIGURATION	The Terminal is restricted due to an unsupported configuration (for example, configuring the rollover option).
staticint	CAUSE_UNSUPPORTED_PROTOCOL	The Terminal in the control list is using a protocol that Cisco Unified JTAPI does not support.
staticint	CAUSE_USER_RESTRICTED	The Terminal or Address is marked as restricted.

Interface	Field	Description
staticint	ID	None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

None

CiscoRouteAddress

This interface is deprecated and has not been implemented.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.Address, javax.telephony.callcenter.RouteAddress

Declaration

public interface CiscoRouteAddress extends javax.telephony.callcenter.RouteAddress

Fields

None

Inherited Fields

From Interface javax.telephony.callcenter.RouteAddress

ALL_ROUTE_ADDRESS

Methods

Table 148: Methods in CiscoRouteAddress

Interface	Method	Description
void	registerRouteCallback (javax.telephony.callcenter.RouteCallbackrouteCallback, booleandisableAutoRehoming)	Deprecated **Throws** javax.telephony.ResourceUnavailableException javax, telephony.MethodNotSupportedException

Inherited Methods

From Interface javax.telephony.callcenter.RouteAddress

cancelRouteCallback, getActiveRouteSessions, getRouteCallback, registerRouteCallback

From Interface javax.telephony.Address

addCallObserver, addObserver, getAddressCapabilities, getCallObservers, getCapabilities, getConnections, getName, getObservers, getProvider, getTerminals, removeCallObserver, removeObserver

Related Documentation

None

CiscoRouteEvent

The CiscoRouteEvent interface extends the RouteEvent interface with additional Cisco-specific capabilities. Applications can use the getCallingPartyIpAddr method to obtain the IP address of the calling party device.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Added the method getCallingPartyIpAddr_v6().

Superinterfaces

javax.telephony.callcenter.events.RouteEvent, javax.telephony.callcenter.events.RouteSessionEvent

Declaration

public interface CiscoRouteEvent extends javax.telephony.callcenter.events.RouteEvent

Fields

None

Inherited Fields

From Interface javax.telephony.callcenter.events.RouteEvent

SELECT_ACD, SELECT_EMERGENCY, SELECT_LEAST_COST, SELECT_NORMAL, SELECT_USER_DEFINED

Methods

Table 149: Methods in CiscoRouteEvent

Interface	Method	Description
java.net.InetAddress	getCallingPartyIpAddr_v6()	Returns the IPv6 address of the calling party. If the IP address is not available, this method returns an InetAddress with the IP address 0::0 and a null host name. Printing this object yields a string representation of "null/0::0". Returns: InetAddress.
java.net.InetAddress	getCallingPartyIpAddr()	Returns the IP address of the calling party. If the IP address is not available, this method returns an InetAddress with the IP address 0.0.0.0 and a null host name. Printing this object yields a string representation of "null/0.0.0.0".

Inherited Methods

From Interface javax.telephony.callcenter.events.RouteEvent

getCallingAddress, getCallingTerminal, getCurrentRouteAddress, getRouteSelectAlgorithm, getSetupInformation

From Interface javax.telephony.callcenter.events.RouteSessionEvent

getRouteSession

Related Documentation

None

CiscoRouteSession

The CiscoRouteSession interface supports application access to the underlying call that is associated with a RouteSession. Also, this interface exposes various internal ERRORs for RouteEndEvent.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.
11.5(1)	Added route selection with deviceName

Superinterfaces

javax.telephony.callcenter.RouteSession

Declaration

public interface CiscoRouteSession extends javax.telephony.callcenter.RouteSession

Fields

Table 150: Fields in CiscoRouteSession

Interface	Field	Description
static final int	ERROR_ROUTESELECT_TIMEOUT	Each routeEvent() or reRouteEvent() that is sent starts a timer for the application to respond with a routeSelect() or endRoute(). The default value of this timer is 5 seconds. Should the application not respond within this time, the system calls an endRoute with this error.
static final int	ERROR_NO_CALLBACK	Because there is no default route mechanism in place, if there is no callback registered for this application, the system calls an endRoute with this error.
static final int	ERROR_INVALID_STATE	If an internal InvalidStateException occurred, or some preconditions or postconditions were not met during routing, the system calls endRoute with this error.
static final int	DEFAULT_SEARCH_SPACE	This indicates that the redirect should be done by using the search space that is the default for the implementation. The default is to use the caller search space.
static final int	CALLINGADDRESS_SEARCH_SPACE	This indicates that the redirect should be done by using the search space of the calling address.

Interface	Field	Description
static final int	ROUTEADDRESS_SEARCH_SPACE	This indicates that the redirect should be done by using the search space of the route point address.
static final int	DONOT_RESET_ORIGINALCALLED	This is a parameter value for the PreferedOriginalCalled option; it specifies not to reset OriginalCalled.
static final int	RESET_ORIGINALCALLED	This is a parameter value for PreferedOriginalCalled Option; if the value of preferedOriginalCalledOption is set to this, it will reset the OriginalCalled to preferedOriginalCalledNumber.
static final int	CAUSE_CTIERR_FAC_CMC_REASON_FAC_NEEDED	This constant returned by RouteSession.getCause() indicates that the routeSelectedElement in the selectRoute does not contain the required FAC code.
static final int	CAUSE_CTIERR_FAC_CMC_REASON_CMC_NEEDED	This constant returned by RouteSession.getCause() indicates that the routeSelectedElement in the selectRoute does not contain the required CMC code.
static final int	CAUSE_CTIERR_FAC_CMC_REASON_FAC_CMC_NEEDED	This constant returned by RouteSession.getCause() indicates that the routeSelectedElement in the selectRoute does not contain the required FAC and CMC codes.
static final int	CAUSE_CTIERR_FAC_CMC_REASON_FAC_INVALID	This constant returned by RouteSession.getCause() indicates that the routeSelectedElement in the selectRoute contains an invalid FAC code.

Interface	Field	Description
static final int	CAUSE_CTIERR_FAC_CMC_REASON_CMC_INVALID	This constant returned by RouteSession.getCause() indicates that the routeSelectedElement in the selectRoute contains an invalid CMC code.

Inherited Fields

From Interface javax.telephony.callcenter.RouteSession

CAUSE_INVALID_DESTINATION, CAUSE_NO_ERROR, CAUSE_PARAMETER_NOT_SUPPORTED, CAUSE_ROUTING_TIMER_EXPIRED, CAUSE_STATE_INCOMPATIBLE, CAUSE_UNSPECIFIED_ERROR, ERROR_RESOURCE_BUSY, ERROR_RESOURCE_OUT_OF_SERVICE, ERROR_UNKNOWN, RE_ROUTE, ROUTE, ROUTE_CALLBACK_ENDED, ROUTE_END, ROUTE_USED

Methods

Table 151: Methods in CiscoRouteSession

Interface	Method	Description
javax.telephony.Call	getCall()	Returns the call associated with this RouteSession.
void	selectRoute(java.lang. String[]routeSelected, intcallingSearchSpace)	Overloads the selectRoute method in the RouteSession interface to allow applications to specify a calling search space to use when the call is redirected to the route destination. **Parameters** javax.telephony. MethodNotSupportedException **Throws** • callingSearchSpace—One of CiscoRouteSession. DEFAULT_SEARCH_SPACE; CiscoRouteSession. CALLINGADDRESS_SEARCH_SPACE; or CiscoRouteSession. ROUTEADDRESS_SEARCH_SPACE. • routeSelected—A list of possible destinations for the call.

Interface	Method	Description
void	selectRoute(java.lang. String[]routeSelected, intcallingSearchSpace, java.lang. String[]modifyingCallingNumber)	

Interface	Method	Description
		Selects one or more possible routing destinations for a call with a modified calling number. This method takes as an argument that is a string array of destination telephone address names, modifyingCallingNumber, arranged in priority order. The highest-priority destination is the first element in the specified array and routing is attempted with this destination first with the corresponding element of modifying calling number. If modifiedCallingNumber is null for an element, the calling number is not modified if a call is routed to that particular routeSelected element. The system attempts to use the specified destination addresses in order until the system successfully selects a destination. The system delivers a RouteUsedEvent to the application when it routes the call to that destination. **Pre-conditions** • this.getRouteAddress().getProvider().getState() = = • Provider.IN_SERVICE this.getState() = = RouteSession.ROUTE or RouteSession.RE_ROUTE **Post-conditions** • this.getRouteAddress().getProvider().getState() = = • Provider.IN_SERVICE this.getState() = = • RouteSession.ROUTE_USED (if the Call was successfully routed.) A RouteUsedEvent gets delivered for this RouteSession if a successful destination was selected. **Parameters** • routeSelected—Possible destinations for callcallingSearchSpace can be CiscoRouteSession.DEFAULT_SEARCH_SPACE; CiscoRouteSession. CALLINGADDRESS_SEARCH_SPACE; or CiscoRouteSession. ROUTEADDRESS_SEARCH_SPACE. • modifyingCallingNumber—An array of elements for which the application wants to modify the calling number when the call reaches the routeSelected element.

Interface	Method	Description
		Throws • com.cisco.jtapi. MethodNotSupportedExceptionImpl (The implementation does not support routing.) • javax.telephony. PrivilegeViolationException (The user does not have the necessary privileges to use this method.) • javax.telephony.MethodNotSupportedException selectRoute

Interface	Method	Description
void	selectRoute(java.lang. String[]routeSelected, intcallingSearchSpace, java.lang. String[]preferedOriginalCalledNumber, int[]preferedOriginalCalledOption	

Interface	Method	Description
		Selects one or more possible destinations for routing the Call. This method takes as an argument a string array of destination telephone address names, in prioritized order, and a string array for the PreferredOriginalCalled number. PreferedOriginalCalled number gets selected based on the index of the destination telephone names array. If the index corresponding to the destination array is not found in the PreferedOriginalCalled number array, preferedOriginalCalled gets set to the destination. The highest-priority destination is the first element in the given array, and the system attempts to route with this destination first. The system attempts each given destination address in succession until the call gets successfully routed. The system delivers a RouteUsedEvent event to the application when a successful routing destination has been selected and the Call has been routed to that destination. **Pre-conditions** • this.getRouteAddress().getProvider().getState() = = • Provider.IN_SERVICE this.getState() = = RouteSession.ROUTE or • RouteSession.RE_ROUTE **Post-conditions** • this.getRouteAddress().getProvider().getState() = = • Provider.IN_SERVICE this.getState() = = • RouteSession.ROUTE_USED (if the Call was successfully routed.) A RouteUsedEvent gets delivered for this RouteSession if a successful destination was selected. **Parameters** • routeSelected—Possible destinations for the call. • preferedOriginalCalledNumber—List with each item corresponding to a route at the matching array index in the routeSelected list. • preferedOriginalCalledOption—List of options, each corresponding to routeSelected list. The option specifies whether to set OriginalCalled to preferedOriginalCalledNumber. The option values

Interface	Method	Description
		are CiscoRouteSession. DONOT_RESET_ORIGINALCALLED and CiscoRouteSession. RESET_ORIGINALCALLED. If the value is unspecified or null, the default is CiscoRouteSession. DONOT_RESET_ORIGINALCALLED. **Throws** com.cisco.jtapi. MethodNotSupportedExceptionImpl (The implementation does not support routing.) javax.telephony. PrivilegeViolationException (The user does not have the necessary privileges to use this method.) javax.telephony. MethodNotSupportedException selectRoute

Interface	Method	Description
void	selectRoute(java.lang. String[]routeSelected, intcallingSearchSpace, java.lang. String[]modifyingCallingNumber, java.lang. String[]preferedOriginalCalledNumber, int[]preferedOriginalCalledOption, java.lang. String[]facCode, java.lang. String[]cmcCode)	

Interface	Method	Description
		Selects one or more possible routing destinations. It takes as arguments a string array of: • destination telephone address names, in prioritized order • PreferredOriginalCalled numbers • FACs • CMCs The system selects the PreferedOriginalCalled number corresponding to the index of the destination telephone name array. If the index is not found in the PreferedOriginalCalled number array, the preferedOriginalCalled gets set to the destination. The highest priority destination is the first element in the specified array, and the system attempts to route the call to that destination first. The system attempts the specified destination addresses in order, until the call gets routed successfully. The system delivers a RouteUsedEvent event to the application when the system has selected a successful routing destination and routed the call to that destination. **Pre-conditions** • this.getRouteAddress().getProvider().getState() = = • Provider.IN_SERVICE this.getState() = = RouteSession.ROUTE • RouteSession.RE_ROUTE **Post-conditions** • this.getRouteAddress().getProvider().getState() = = • Provider.IN_SERVICE this.getState() = = • RouteSession.ROUTE_USED (if the call was successfully routed). A RouteUsedEvent gets delivered for this RouteSession if a successful destination was selected. **Parameters** • routeSelected—List of possible destinations. • preferedOriginalCalledNumber—List with each member of corresponding to the route at the same array index in the routeSelected. • list.preferedOriginalCalledOption—List of options,

Interface	Method	Description
		each corresponding to RouteList. The option specifies whether to set OriginalCalled to preferedOriginalCalledNumber. The option values are CiscoRouteSession. DONOT_RESET_ORIGINALCALLED and CiscoRouteSession. RESET_ORIGINALCALLED.If the value is unspecified or null, the default is CiscoRouteSession. DONOT_RESET_ORIGINALCALLED. • modifyingCallingNumber—Array of elements for which the application wants modify the calling number when the call reaches the routeSelected element. If applications do not want to modify the number, a null value for this parameter must be passed by the application. • facCode (Forced Authorization Code [FAC])—If a routeSelected element requires a FAC, the corresponding facCode element must contain that code. If no code is required for a routeSelected element, the application must pass a null value for this parameter. • cmcCode - (Client Matter Code [CMC]) If a routeSelected element requires a CMC, the corresponding cmcCode element must contain that code. If no code is required for a routeSelected element, the application must pass a null value for this parameter. **Throws** • com.cisco.jtapi. MethodNotSupportedExceptionImpl (The implementation does not support routing.) • javax.telephony. PrivilegeViolationException (The user does not have the necessary privileges to use this method.) • javax.telephony. MethodNotSupportedException selectRoute

Interface	Method	Description
void	selectRoute(java.lang. String[]routeSelected, intcallingSearchSpace, java.lang. String[]modifyingCallingNumber, java.lang. String[]preferedOriginalCalledNumber, int[]preferedOriginalCalledOption, java.lang. String[]facCode, java.lang. String[]cmcCode, intfeaturePriority)	

Interface	Method	Description
		Selects one or more possible routing destinations. It takes a string array of: • Destination telephone address names, in prioritized order • PreferredOriginalCalled numbers • FACs • CMCs • Integer priorities The PreferedOriginalCalled number gets selected based on the index of the destination telephone name array. If the index tis not found in the PreferedOriginalCalled number array, preferedOriginalCalled gets set to the destination. The highest-priority destination is the first element in the given array, and the system attempts to route with this destination first. The system tries the successive specified destination addresses until the call gets routed successfully. The system delivers a RouteUsedEvent event to the application when a successful routing destination has been selected and the call has been routed to that destination. **Pre-conditions** • this.getRouteAddress().getProvider().getState() = = • Provider.IN_SERVICE this.getState() = = RouteSession.ROUTE • RouteSession.RE_ROUTE **Post-conditions** • this.getRouteAddress().getProvider().getState() = = • Provider.IN_SERVICE this.getState() = = • RouteSession.ROUTE_USED (if the Call was successfully routed.) **Parameters** routeSelected—Possible destinations for the call. preferedOriginalCalledNumber—List with each element corresponding to the route at the same array index in the routeSelected list. preferedOriginalCalledOption—Options list, each corresponding to RouteList. The option specifies whether

Interface	Method	Description

		to set OriginalCalled to preferedOriginalCalledNumber. The option values are CiscoRouteSession. DONOT_RESET_ORIGINALCALLED and CiscoRouteSession. RESET_ORIGINALCALLED. If the value is unspecified or null, the default is CiscoRouteSession. DONOT_RESET_ORIGINALCALLED.
		• modifyingCallingNumber—Array of elements for which the application would like to modify the calling number when the call reaches the routeselected element. If applications do not want to modify the number, they must pass a null value for this parameter.
		• facCode (Forced Authorization Code [FAC])—If a routeSelected element requires a FAC, the corresponding facCode element must contain that code. If no code is required for a routeSelected element, the application must pass a null value for this parameter.
		• cmcCode (Client Matter Code [CMC])—If a routeSelected element requires a CMC, the corresponding cmcCode element must contain that code. If no code is required for a routeSelected element, the application must pass a null value for this parameter.
		• featurePriority—Sets the feature priority of the call. The application may set CiscoCall. FEATUREPRIORITY_NORMAL if the application does not want to set any specific priority. The featurePriority parameter may be:
		  • CiscoCall. FEATUREPRIORITY_NORMAL
		  • CiscoCall. FEATUREPRIORITY_URGENT
		  • CiscoCall. FEATUREPRIORITY_EMERGENCY
		Throws
		• javax.telephony. PrivilegeViolationException (The user does not have the necessary privileges to use this method.)
		• com.cisco.jtapi. MethodNotSupportedExceptionImpl (The implementation does not support routing.)
		• javax.telephony. MethodNotSupportedException

Interface	Method	Description
void	selectRoute(java.lang. String[]routeSelected, int[]callingSearchSpace, java.lang. String[]modifyingCallingNumber, java.lang. String[]preferedOriginalCalledNumber, int[]preferedOriginalCalledOption, java.lang. String[]facCode, java.lang. String[]cmcCode, int[]featurePriority)	

Interface	Method	Description
		Selects one or more possible routing destinations. It takes a string array of: • destination telephone address names, in prioritized order • calling search spaces • modifyingCallingNumbers • PreferredOriginalCalled numbers • FACs • CMCs • feature priorities The PreferedOriginalCalled number gets selected based on the index of the destination telephone name array. If the index tis not found in the PreferedOriginalCalled number array, preferedOriginalCalled gets set to the destination. The highest-priority destination is the first element in the given array, and the system attempts to route with this destination first. The system tries the successive specified destination addresses until the call gets routed successfully. The system delivers a RouteUsedEvent event to the application when a successful routing destination has been selected and the call has been routed to that destination. **Pre-conditions** • this.getRouteAddress().getProvider().getState() = = • Provider.IN_SERVICE this.getState() = = RouteSession.ROUTE • RouteSession.RE_ROUTE **Post-conditions** • this.getRouteAddress().getProvider().getState() = = • Provider.IN_SERVICE this.getState() = = • RouteSession.ROUTE_USED (if the call was successfully routed.) **Parameters** • routeSelected—List of possible destinations for the call. • callingSearchSpace—For each route selected; can

Interface	Method	Description
		be CiscoRouteSession. DEFAULT_SEARCH_SPACE, CiscoRouteSession. CALLINGADDRESS_SEARCH_SPACE, or CiscoRouteSession. ROUTEADDRESS_SEARCH_SPACE. • preferedOriginalCalledNumber—List with each element corresponding to the route at the same array index in the routeSelected list. • preferedOriginalCalledOption—Options list, each corresponding to RouteList. The option specifies whether to set OriginalCalled to preferedOriginalCalledNumber. The option values are CiscoRouteSession. DONOT_RESET_ORIGINALCALLED and CiscoRouteSession. RESET_ORIGINALCALLED. If the value is unspecified or null, the default is CiscoRouteSession. DONOT_RESET_ORIGINALCALLED. • modifyingCallingNumber—Elements array for which the application would like to modify the calling number when the call reaches the routeselected element. If applications do not want to modify the number, they must pass a null value for this parameter. • facCode (Forced Authorization Code [FAC])—If a routeSelected element requires a FAC, the corresponding facCode element must contain that code. If no code is required for a routeSelected element, the application must pass a null value for this parameter. • cmcCode (Client Matter Code [CMC])—If a routeSelected element requires a CMC, the corresponding cmcCode element must contain that code. If no code is required for a routeSelected element, the application must pass a null value for this parameter. • featurePriority—For each route selected, the feature priority can be set. The application may set CiscoCall. FEATUREPRIORITY_NORMAL if the application does not want to set any specific priority. The featurePriority parameter may be CiscoCall. FEATUREPRIORITY_NORMAL, CiscoCall. FEATUREPRIORITY_URGENT, or CiscoCall.FEATUREPRIORITY_EMERGENCY **Throws**

Interface	Method	Description
		• javax.telephony. PrivilegeViolationException (The user does not have the necessary privileges to use this method.) • com.cisco.jtapi. MethodNotSupportedExceptionImpl (The implementation does not support routing.) • javax.telephony. MethodNotSupportedException
void	selectRoute(String[] routeSelected, int[] callingSearchSpace, String[] modifyingCallingNumber,String[] preferedOriginalCalledNumber, int[] preferedOriginalCalledOption, String[] facCode, String[] cmcCode,int[] featurePriority, byte[][] applicationXMLData, String[] deviceName	This method is similar to the above method, but takes an array of destination device names, in a prioritized order. This order of device names corresponds to the order of destinations provided in route selected. This method takes a string array of: • destination telephone address names, in prioritized order • calling search spaces • modifyingCallingNumbers • PreferredOriginalCalled numbers • FACs • CMCs • feature priorities • Application XML • device Names

Inherited Methods

From Interface javax.telephony.callcenter.RouteSession

endRoute, getCause, getRouteAddress, getState, selectRoute

Related Documentation

See Constant Field Values.

CiscoRouteTerminal

A CiscoRouteTerminal is a special kind of CiscoTerminal that allows applications to terminate RTP media streams. Unlike a CiscoTerminal, a CiscoRouteTerminal does not represent a physical telephony endpoint,

which is observable and controllable in a third-party manner. Instead, a CiscoRouteTerminal is a logical telephony endpoint that may be associated with any application that wants to route calls and terminate media. Unlike a CiscoMediaTerminal, a CiscoRouteTerminal can have multiple active calls at the same time. Typically, applications use CiscoRouteTerminals to queue calls until an agent is available to service them.

Note CiscoRouteTerminals are CTI Route Points on Cisco Unified Communications Manager.

Terminating media is a three-step process as follows:

1. The application registers its media capabilities with this Terminal by using the CiscoRouteTerminal.register method.
2. The application adds an observer that implements the CiscoTerminalObserver interface by using the Terminal.addObserver method.
3. The application must call addCallObserver on the CiscoRouteTerminal or the CiscoRouteAddress to receive and answer calls.

Applications will receive a CiscoMediaOpenLogicalChannelEv for each call or whenever media is stopped and needs to be reestablished. Applications must supply an IP address and port number by using the setRTPParams method on CiscoRouteTerminal.

Note **Important**—All applications written for or prior to CiscoJtapiClient Release 1.4 must be modified to register with CiscoRouteTerminal.NO_MEDIA_TERMINATION type if the applications are not interested in media termination.

Multiple applications can register with same RoutePoint as long as they are registered with the same media capabilities and registration type. All applications, if registered with CiscoRouteTerminal.DYNAMIC_MEDIA_REGISTRATION, will receive CiscoMediaOpenLogicalChannelEv when they add a callObserver, but only one application will be able to invoke setRTPParams.

Applications that are interested in media termination must add a CallObserver on the RouteAddress or on the CiscoRouteTerminals. Applications must not register with Dynamic type and add a registerRouteCallBack. Applications should only use registerRouteCallBack if they are not interested in media termination. Applications must not add a registerRouteCallBack and a callObserver at the same time.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Added these modes to the activeAddressingMode: • CiscoTerminal.IP_ADDRESSING_MODE_IPv6 • CiscoTerminal.IP_ADDRESSING_MODE_IPv4_v6

Superinterfaces

CiscoObjectContainer, CiscoTerminal, javax.telephony.Terminal

Declaration

public interface CiscoRouteTerminal extends CiscoTerminal

Fields

Table 152: Fields in CiscoRouteTerminal

Interface	Field	Description
staticint	DYNAMIC_MEDIA_REGISTRATION	Applications that are interested in media termination need to register with this type and pass in the capabilities that the application supports in the registration request.
staticint	NO_MEDIA_REGISTRATION	Applications that are not interested in media termination need to register with this type and pass in a null value for CiscoMediaCapability in the registration request. If registrationType is CiscoRouteTerminal. NO_MEDIA_REGISTRATION, the application cannot terminate media and can use the CiscoRouteTerminal for call routing.

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoTerminal

ASCII_ENCODING, DEVICESTATE_ACTIVE, DEVICESTATE_ALERTING, DEVICESTATE_HELD, DEVICESTATE_IDLE, DEVICESTATE_UNKNOWN, DEVICESTATE_WHISPER, DND_OPTION_CALL_REJECT, DND_OPTION_NONE, DND_OPTION_RINGER_OFF, IN_SERVICE, IP_ADDRESSING_MODE_IPV4, IP_ADDRESSING_MODE_IPV4_V6, IP_ADDRESSING_MODE_IPV6, IP_ADDRESSING_MODE_UNKNOWN, IP_ADDRESSING_MODE_UNKNOWN_ANATRED, NOT_APPLICABLE, OUT_OF_SERVICE, UCS2UNICODE_ENCODING, UNKNOWN_ENCODING

Methods

Table 153: Methods in CiscoRouteTerminal

Interface	Method	Description
void	register(CiscoMediaCapability[]capabilities, intregistrationType)	Registers a Terminal with specified CiscoMediaCapabilities and register type. The Provider must be in the Provider.IN_SERVICE state. Returns successfully when the CiscoRouteTerminal is registered. **Parameters** • capabilities—List of RTP encodings that the application supports for this Terminal. If the application is not interested in media termination, you may pass in a null value. • registrationType—Either CiscoRouteTerminal .DYNAMIC_MEDIA_REGISTRATION or CiscoRouteTerminal .NO_MEDIA_REGISTRATION. If registrationType is CiscoRouteTerminal .NO_MEDIA_REGISTRATION, the application cannot terminate media and can use the CiscoRouteTerminal for call routing. If registrationType is CiscoRouteTerminal .DYNAMIC_MEDIA_REGISTRATION, the application can terminate media and can have multiple active calls. This registrationType indicates that the application will supply the IP address and port dynamically for each call. Applications registering with this type receive a CiscoMediaOpenLogicalChannelEv for each call and must supply the IP address and port number by using the setRTPParams method on the CiscoRouteTerminal . **Throws** • CiscoRegistrationException

Interface	Method	Description
void	register(CiscoMediaCapability[]capabilities, intregistrationType, int[]algorithmIDs)	Registers a Terminal with the specified CiscoMediaCapabilities, registrationType, and supported SRTP algorithms. The Provider must be in the Provider.IN_SERVICE state. This method returns successfully when the CiscoRouteTerminal is registered. **Parameters** • capabilities—List of RTP encodings that the application supports for this Terminal. If the application is not interested in media termination, you may pass in a null value. • registrationType—Either CiscoRouteTerminal .DYNAMIC_MEDIA_REGISTRATION or CiscoRouteTerminal .NO_MEDIA_REGISTRATION. If registrationType is CiscoRouteTerminal .NO_MEDIA_REGISTRATION, the application cannot terminate media and can use the CiscoRouteTerminal for call routing. Other parameters in the method are ignored. If registrationType is CiscoRouteTerminal .DYNAMIC_MEDIA_REGISTRATION, the application can terminate media and can have multiple active calls. This registrationType indicates that the application will supply the IP address and port dynamically for each call. Applications registering with this type receive a CiscoMediaOpenLogicalChannelEv for each call and must supply the IP address and port number by using the setRTPParams method. • algorithmIDs—List of SRTP algorithms that the application supports for this Terminal. To use this, the application must have the TLS Link and SRTP Enabled flag enabled. AlgorithmIDs must be one of CiscoMediaEncryptionSupportedAlgorithms. **Throws** • javax.telephony.PrivilegeViolationException (The application tried to use the method, but is not authorized to use it.) • CiscoRegistrationException

Interface	Method	Description
void	register(CiscoMediaCapability[]capabilities, intregistrationType, int[]algorithmIDs, intactiveAddressingMode)	

Interface	Method	Description
		The CiscoRouteTerminal must be in the CiscoTerminal.UNREGISTERED state and its Provider must be in the Provider.IN_SERVICE state. The successful effect of this method is to register the RouteTerminal. Registers a Terminal with specified CiscoMediaCapabilities and register type and supported SRTP Algorithms. If registrationType is CiscoRouteTerminal .NO_MEDIA_REGISTRATION, application cannot terminate media and can use route point for call routing purpose. Other parameters in the method are ignored. If registration Type is CiscoRouteTerminal .DYNAMIC_MEDIA_REGISTRATION, then app can terminate media and can have multiple active calls. This Indicates that application is interested in supplying ipAddress and port dynamically for each call. Applications registering with this type will receive CiscoMediaOpenLogicalChannelEv for each call and will have to supply ipAddress and port number using setRTPParams method on CiscoRouteTerminal . **Method arguments** Capabilities indicates the type of RTP encodings that the application is willing to support for this Terminal. If application is not intersted in media termination, it may pass in null value registrationType may be CiscoRouteTerminal .NO_MEDIA_REGISTRATION or CiscoRouteTerminal .DYNAMIC_MEDIA_REGISTRATION. Supported Algorithms may be the SRTP Algorithms that application supports for this terminal. In order to use this, application need to have TLS Link and SRTP Enabled flag enabled. PrivilegeViolationException is thrown if app is not authorized to use this method. **Post-condition** This method returns successfully when the CiscoRouteTerminal is registered. **Parameters** • capabilities—List of RTP encodings supported by this terminal. • registrationType—CiscoRouteTerminal .DYNAMIC_MEDIA_REGISTRATION or CiscoRouteTerminal .NO_MEDIA_REGISTRATION

Interface	Method	Description

		• algorithmIDs—List of supported SRTP algorithms. AlgorithmIDs may only be one of CiscoMediaEncryptionSupportedAlgorithms.
		• activeAddressingMode—IP Addressing mode in which application intends to register this CiscoRouteTerminal . The modes can be:
		  • CiscoTerminal.IP_ ADDRESSING_MODE_IPv4
		  • CiscoTerminal.IP_ ADDRESSING_MODE_IPv6
		  • CiscoTerminal.IP_ ADDRESSING_MODE_IPv4_v6
		Throws
		• CiscoRegistrationException
		• javax.telephony.PrivilegeViolationException

void	unregister()	The CiscoRouteTerminal must be registered and its Provider must be in the Provider.IN_SERVICE state. The successful effect of this method is to unregister the CiscoRouteTerminal . **Post-condition** • This method returns successfully when the MediaTerminal is unregistered. **Throws** • CiscoUnregistrationException

Interface	Method	Description
void	setRTPParams(CiscoRTPHandlertpHandle, CiscoRTPParamsrtpParams)	Applications can set the IP address and RTP Port number to dynamically stream media for a call. To do this, applications must register MediaTerminal or CiscoRouteTeminal by providing only capabilities. Applications must then invoke this method upon receiving CiscoCallOpenLogicalChannelEv on the TerminalObserver. **Parameters** rtpHandle—The handle the application receives in CiscoCallOpenLogicalChannelEvrtpParams. Refer to CiscoRTPParams. **Throws** • javax.telephony.InvalidStateException • javax.telephony.InvalidArgumentException • javax.telephony.PrivilegeViolationException
boolean	isRegistered()	This method returns true if the CiscoMediaTerminal is registered and false otherwise. If the CiscoRouteTerminal is OutOfService, this method returns false; if it is InService, this method returns true. For CTIManager failure cases, this method returns false.
boolean	isRegisteredByThisApp()	This method returns true if this application issued a successful registration request. The registration remains valid even if the device is out-of-service because of a CTIManager failure. This returns true until this application unregisters the device.
int	getIPAddressingMode()	Application can invoke this API to query the IP Addressing Mode of the CiscoRouteTerminal . Addressing mode may be any of the following constants: • CiscoTerminal.IP_ ADDRESSING_IPv4 • CiscoTerminal.IP_ ADDRESSING_IPv6 • CiscoTerminal.IP_ ADDRESSING_IPv4_v6

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoTerminal

createSnapshot, getAltScript, getDeviceState, getDNDOption, getDNDStatus, getEMLoginUsername, getFilter, getLocale, getProtocol, getRegistrationState, getRTPInputProperties, getRTPOutputProperties, getState, getSupportedEncoding, isRestricted, sendData, sendData, setDNDStatus, setFilter, unPark

From Interface javax.telephony.Terminal

addCallObserver, addObserver, getAddresses, getCallObservers, getCapabilities, getName, getObservers, getProvider, getTerminalCapabilities, getTerminalConnections, removeCallObserver, removeObserver

From Interface com.cisco.jtapi.extensions.CiscoObjectContainer

getObject, setObject

Related Documentation

See CiscoTerminal and Constant Field Values CiscoMediaOpenLogicalChannelEv

CiscoRouteUsedEvent

The CiscoRouteUsedEvent event indicates that the RouteSession moved into the RouteSession.ROUTE_USED state and the call terminated at a destination as a result of application routing. This interface extends the RouteUsedEvent interface and gets reported via the RouteCallback interface.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.callcenter.events.RouteSessionEvent, javax.telephony.callcenter.events.RouteUsedEvent

Declaration

public interface CiscoRouteUsedEvent extends javax.telephony.callcenter.events.RouteUsedEvent

Fields

None

Methods

Table 154: Methods in CiscoRouteUsedEvent

Interface	Method	Description
Int	getRouteSelectedIndex()	Returns an array index of the route where the call got routed.

Inherited Methods

From Interface javax.telephony.callcenter.events.RouteUsedEvent

getCallingAddress, getCallingTerminal, getDomain, getRouteUsed

From Interface javax.telephony.callcenter.events.RouteSessionEvent

getRouteSession

Related Documentation

See RouteSession, RouteCallback, and RouteSessionEvent.

CiscoRTPBitRate

The RTPBitRate interface contains constants describing G.723 RTP bit rates. CiscoRTPInputProperties.getBitRate and CiscoRTPOutputProperties.getBitRate return these constants.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoRTPBitRate

Fields

Table 155: Fields in CiscoRTPBitRate

Interface	Fields	Description
staticint	R5_3	This constant is the 5.3k G.723 bit rate.
staticint	R6_4	This constant is the 6.4k G.723 bit rate.

Methods

None

Related Documentation

See CiscoRTPInputProperties.getBitRate(), CiscoRTPOutputProperties.getBitRate()

CiscoRTPHandle

Use the CiscoRTPHandle object to get a call reference with CiscoProvider.getCall(CiscoRTPHandle). This object gets returned in CiscoMediaCallOpenLogicalChannelEv. Pass this handle in the setRTPParams parameter of CiscoMediaTerminal or CiscoRouteTerminal, depending on where the CiscoMediaCallOpenLogicalChannelEv event gets received.

If no call observer was added, or there was no call observer added at the time CiscoMediaCallOpen LogicalChannelEv got sent, CiscoProvider.getCall(CiscoRTPHandle) may return null.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoRTPHandle

Fields

None

Methods

Table 156: Methods in CiscoRTPHandle

Interface	Method	Description
Int	getHandle()	Returns the Cisco Unified Communications Manager CallLeg ID of the call, in integer format.

Related Documentation

None

CiscoRTPInputKeyEv

The CiscoRTPInputKeyEv event interface gives the key information for the encrypted incoming media stream. Applications should set the filter by using CiscoTermEvFilter.setRTPKeyEventsEnabled(true) to get this event via the TerminalObserver.terminalChangedEvent().

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoRTPInputKeyEv extends CiscoTermEv

Fields

Table 157: Fields in CiscoRTPInputKeyEv

Interface	Field	Description
staticint	ID	None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN,

META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 158: Methods in CiscoRTPInputKeyEv

Interface	Method	Description
int	getCiscoMediaEncryptionKeyInfo()	Returns CiscoMediaEncryptionKeyInfo only if the provider is opened with the TLS link and SRTP enabled options set for the application in the Cisco Unified Communications Manager administration. Otherwise, it returns null.
int	getCiscoMediaSecurityIndicator()	Returns the media security indicator, one of the following constants: CiscoMediaSecurityIndicator. MEDIA_ENCRYPTED_KEYS_AVAILABLE CiscoMediaSecurityIndicator. MEDIA_ENCRYPT_USER_NOT_AUTHORIZED CiscoMediaSecurityIndicator. MEDIA_ENCRYPTED_KEYS_UNAVAILABLE
CiscoCallID	getCallID()	Returns a CiscoCallID object if there is already a CiscoCall present when this event is sent. If there is no CiscoCall present, this method returns null. getCallID().getCall() gives the call for which this key applies.
int	getCiscoRTPHandle()	Returns a CiscoRTPHandle object. Applications can get a call reference by using CiscoProvider.getCall. If there is no call observer or there was no call observer when this event got delivered, CiscoProvider.getCall returns null. Returns:CiscoRTPHandle.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See CiscoRTPParams, CiscoMediaSecurityIndicator.

CiscoRTPInputProperties

The CiscoRTPInputProperties interface returns the properties of the media received by the Terminal (the inbound media stream). CiscoRTPInputStartedEv indicates that the media started.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoRTPInputProperties

Fields

None

Methods

Table 159: Methods in CiscoRTPInputProperties

Interface	Method	Description
int	getBitRate()	Returns the media bit rate and can CiscoRTPBitRate.R5_3 or CiscoRTPBitRate.R6_4.
boolean	getEchoCancellation()	Returns True if the application needs to use echo cancellation.
java.net.InetAddress	getLocalAddress()	Returns the address to which media will be directed.
int	getLocalPort()	Returns the port to which media will be directed.
int	getPacketSize()	Returns the packet size, in milliseconds.

Interface	Method	Description
int	getPayloadType()	Returns the payload format, which is one of the following constants: • CiscoRTPPayload.G711ALAW64K • CiscoRTPPayload.G711ALAW56K • CiscoRTPPayload.G711ULAW64K • CiscoRTPPayload.G711ULAW56K • CiscoRTPPayload.G722_64K • CiscoRTPPayload.G722_56K • CiscoRTPPayload.G722_48K • CiscoRTPPayload.G7231 • CiscoRTPPayload.G728 • CiscoRTPPayload.G729 • CiscoRTPPayload.G729ANNEXA • CiscoRTPPayload.ACY_G729AASSN • CiscoRTPPayload.DATA64 • CiscoRTPPayload.DATA56 • CiscoRTPPayload.GSM • CiscoRTPPayload.WIDEBAND_256K

Related Documentation

See CiscoRTPPayload and CiscoRTPBitRate.

CiscoRTPInputStartedEv

The CiscoRTPInputStartedEv event indicates the start of incoming media.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoRTPInputStartedEv extends CiscoTermEv

Fields

Table 160: Fields in CiscoRTPInputStartedEv

Interface	Field	Description
staticint	ID	None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 161: Methods in CiscoRTPInputStartedEv

Interface	Method	Description
CiscoCallID	getCallID()	Returns CiscoCallID.
CiscoRTPHandle	getCiscoRTPHandle()	Returns a CiscoRTPHandle object.
int	getMediaConnectionMode()	Returns a CiscoMediaConnectionMode.
int	getRTPInputProperties()	Returns CiscoRTPInputProperties, which gives the characteristics of the incoming media.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values, CiscoRTPInputProperties, CiscoCallID, CiscoRTPParams, and CiscoMediaConnectionMode.

CiscoRTPInputStoppedEv

The CiscoRTPInputStoppedEv event indicates that the incoming media stream has stopped.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoRTPInputStoppedEv extends CiscoTermEv

Fields

Table 162: Fields in CiscoRTPInputStoppedEv

Interface	Field	Description
staticint	ID	None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 163: Methods in CiscoRTPInputStoppedEv

Interface	Method	Description
CiscoCallID	getCallID()	Returns CiscoCallID. CiscoRTPInputStartedEv applies to CiscoCallID.getCall().
CiscoRTPHandle	getCiscoRTPHandle()	Returns CiscoRTPHandle object. Applications can get call reference using CiscoProvider.getCall If there is no callobserver or there was no callobserver when this event is delivered, then CiscoProvider.getCall may return null.
int	getMediaConnectionMode()	Returns a CiscoMediaConnectionMode with one of the following values for mediaMode: • CiscoMediaConnectionMode. RECEIVE_ONLY (one-way media, receive only) • CiscoMediaConnectionMode. TRANSMIT_AND_RECEIVE: (two-way media) In general, you should never get an event with mode NONE; however, if that happens, applications should ignore the event and log an error.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values, CiscoMediaConnectionMode, CiscoCallID, and CiscoRTPParams.

CiscoRTPOutputKeyEv

The CiscoRTPOutputKeyEv event gives the key information for the encrypted outgoing (transmitted) media stream. Applications set the filter by using CiscoTermEvFilter.setRTPKeyEventsEnabled(true) to get this event via the TerminalObserver.terminalChangedEvent().

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoRTPOutputKeyEv extends CiscoTermEv

Fields

Table 164: Fields in CiscoRTPOutputKeyEv

Interface	Field	Description
staticint	ID	None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 165: Methods in CisctoRTPOutputKeyEv

Interface	Method	Description
CiscoCallID	getCallID()	Returns a CiscoCallID object if there is already a CiscoCall present when this event is sent. If there is no CiscoCall present, this method returns null.
CiscoMediaEncryptionKeyInfo	getCiscoMediaEncryptionKeyInfo()	Returns CiscoMediaEncryptionKeyInfo only if the provider is opened with the TLS link and SRTP enabled options set for the application in Cisco Unified Communications Manager administration. Otherwise, it will return null.
int	getCiscoMediaSecurityIndicator()	Returns media security indicator, one of the following constants: • CiscoMediaSecurityIndicator. MEDIA_ENCRYPTED_KEYS_AVAILABLE • CiscoMediaSecurityIndicator. MEDIA_ENCRYPT_USER_NOT_AUTHORIZED • CiscoMediaSecurityIndicator. MEDIA_ENCRYPTED_KEYS_UNAVAILABLE

Interface	Method	Description
CiscoRTPHandle	getCiscoRTPHandle()	Returns a CiscoRTPHandle object. Applications can get a call reference by using CiscoProvider.getCall. If there is no call observer or there was no call observer when this event is delivered, CiscoProvider.getCall may return null.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values, CiscoRTPParams, and CiscoMediaSecurityIndicator.

CiscoRTPOutputProperties

The CiscoRTPOutputProperties interface gives the properties of the media transmitted by the terminal. CiscoRTPOutputStartedEv indicates that the media has started.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoRTPOutputProperties

Fields

None

Methods

Table 166: Methods in CiscoRTPOutputProperties

Interface	Method	Description
int	getBitRate()	Returns the media bit rate, one of the following constants: • CiscoRTPBitRate.R5_3 • CiscoRTPBitRate.R6_4
int	getMaxFramesPerPacket()	Returns the maximum number of frames to send per packet.
int	getPacketSize()	Returns the packet size, in milliseconds.
int	getPayloadType()	Returns the payload format, which is one of the following constants: • CiscoRTPPayload.NONSTANDARD • CiscoRTPPayload.G711ALAW64K • CiscoRTPPayload.G711ALAW56K • CiscoRTPPayload.G711ULAW64K • CiscoRTPPayload.G711ULAW56K • CiscoRTPPayload.G722_64K • CiscoRTPPayload.G722_56K • CiscoRTPPayload.G722_48K • CiscoRTPPayload.G7231 • CiscoRTPPayload.G728 • CiscoRTPPayload.G729 • CiscoRTPPayload.G729ANNEXA • CiscoRTPPayload.IS11172AUDIOCAP • CiscoRTPPayload.IS13818AUDIOCAP • CiscoRTPPayload.ACY_G729AASSN • CiscoRTPPayload.DATA64 • CiscoRTPPayload.DATA56 • CiscoRTPPayload.GSM • CiscoRTPPayload.ACTIVEVOICE • CiscoRTPPayload.WIDEBAND_256K
int	getPrecedenceValue()	Returns the precedence value.
java.net. InetAddress	getRemoteAddress()	Returns the address to which media is to be transmitted.
int	getRemotePort()	Returns the port to which media is to be transmitted.
boolean	getSilenceSuppression()	Returns false if Cisco Unified Communication Manager service parameter “Silence Suppression” is set to False or True otherwise.

Related Documentation

See CiscoRTPBitRate.

CiscoRTPOutputStartedEv

The CiscoRTPOutputStartedEv event interface indicates the start of media transmission.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoRTPOutputStartedEv extends CiscoTermEv

Fields

Table 167: Fields in CiscoRTPOutputStartedEv

Interface	Field	Description
staticint	ID	None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 168: Methods in CiscoRTPOutputStartedEv

Interface	Method	Description
CiscoRTPOutputProperties	getRTPOutputProperties()	Returns the RTP output properties.
CiscoCallID	getCallID()	Returns CiscoCallID. CiscoRTPOutputStartedEv applies to CiscoCallID.getCall().
CiscoRTPHandle	getCiscoRTPHandle()	Returns a CiscoRTPHandle object. Applications can get a call reference by using CiscoProvider.getCall. If there is no call observer or there was no call observer when this event is delivered, CiscoProvider.getCall may return null.
int	getMediaConnectionMode()	Returns a CiscoMediaConnectionMode with one of the following values for mediaMode: • CiscoMediaConnectionMode. TRANSMIT_ONLY (one-way media; transmit only) • CiscoMediaConnectionMode. TRANSMIT_AND_RECEIVE (two-way media) **Note** In general, you should never get an event with mode NONE; however, if that happens, applications should ignore the event and log an error.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values, CiscoCallID, and CiscoRTPParams.

CiscoRTPOutputStoppedEv

The CiscoRTPOutputStoppedEv event indicates that the media transmission stopped.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoRTPOutputStoppedEv extends CiscoTermEv

Fields

Table 169: Fields in CiscoRTPOutputStoppedEv

Interface	Field	Description
staticint	ID	None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 170: Methods in CiscoRTPOutputStoppedEv

Interface	Method	Description
CiscoCallID	getCallID()	Returns CiscoCallID. CiscoRTPOutputStoppedEv applies to CiscoCallID.getCall().
CiscoRTPHandle	getCiscoRTPHandle()	Returns a CiscoRTPHandle object. Applications can get a call reference by using CiscoProvider.getCall. If there is no call observer or there was no call observer when this event is delivered, CiscoProvider.getCall may return null.
int	getMediaConnectionMode()	• Returns CiscoMediaConnectionMode with one of the following values: • CiscoMediaConnectionMode. TRANSMIT_ONLY (one-way media; transmit) • onlyCiscoMediaConnectionMode. TRANSMIT_AND_RECEIVE (two-way media) **Note** In general, you should never get an event with mode NONE; however, if that happens, applications should ignore the event and log an error.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values, CiscoCallID, CiscoRTPParams, and CiscoMediaConnectionMode.

CiscoRTPOutputKeyEv

The CiscoRTPOutputKeyEv event gives the key information for the encrypted outgoing (transmitted) media stream. Applications set the filter by using CiscoTermEvFilter.setRTPKeyEventsEnabled(true) to get this event via the TerminalObserver.terminalChangedEvent().

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoRTPOutputKeyEv extends CiscoTermEv

Fields

Table 171: Fields in CiscoRTPOutputKeyEv

Interface	Field	Description
staticint	ID	None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN,

META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 172: Methods in CisctoRTPOutputKeyEv

Interface	Method	Description
CiscoCallID	getCallID()	Returns a CiscoCallID object if there is already a CiscoCall present when this event is sent. If there is no CiscoCall present, this method returns null.
CiscoMediaEncryptionKeyInfo	getCiscoMediaEncryptionKeyInfo()	Returns CiscoMediaEncryptionKeyInfo only if the provider is opened with the TLS link and SRTP enabled options set for the application in Cisco Unified Communications Manager administration. Otherwise, it will return null.
int	getCiscoMediaSecurityIndicator()	Returns media security indicator, one of the following constants: • CiscoMediaSecurityIndicator. MEDIA_ENCRYPTED_KEYS_AVAILABLE • CiscoMediaSecurityIndicator. MEDIA_ENCRYPT_USER_NOT_AUTHORIZED • CiscoMediaSecurityIndicator. MEDIA_ENCRYPTED_KEYS_UNAVAILABLE
CiscoRTPHandle	getCiscoRTPHandle()	Returns a CiscoRTPHandle object. Applications can get a call reference by using CiscoProvider.getCall. If there is no call observer or there was no call observer when this event is delivered, CiscoProvider.getCall may return null.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values, CiscoRTPParams, and CiscoMediaSecurityIndicator.

CiscoRTPOutputProperties

The CiscoRTPPutputProperties interface gives the properties of the media transmitted by the terminal. CiscoRTPOutPutStartedEv indicates that the media has started.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoRTPOutputProperties

Fields

None

Methods

Table 173: Methods in CiscoRTPOutputProperties

Interface	Method	Description
int	getBitRate()	Returns the media bit rate, one of the following constants: • CiscoRTPBitRate.R5_3 • CiscoRTPBitRate.R6_4
int	getMaxFramesPerPacket()	Returns the maximum number of frames to send per packet.
int	getPacketSize()	Returns the packet size, in milliseconds.

Interface	Method	Description
int	getPayloadType()	Returns the payload format, which is one of the following constants: • CiscoRTPPayload.NONSTANDARD • CiscoRTPPayload.G711ALAW64K • CiscoRTPPayload.G711ALAW56K • CiscoRTPPayload.G711ULAW64K • CiscoRTPPayload.G711ULAW56K • CiscoRTPPayload.G722_64K • CiscoRTPPayload.G722_56K • CiscoRTPPayload.G722_48K • CiscoRTPPayload.G7231 • CiscoRTPPayload.G728 • CiscoRTPPayload.G729 • CiscoRTPPayload.G729ANNEXA • CiscoRTPPayload.IS11172AUDIOCAP • CiscoRTPPayload.IS13818AUDIOCAP • CiscoRTPPayload.ACY_G729AASSN • CiscoRTPPayload.DATA64 • CiscoRTPPayload.DATA56 • CiscoRTPPayload.GSM • CiscoRTPPayload.ACTIVEVOICE • CiscoRTPPayload.WIDEBAND_256K
int	getPrecedenceValue()	Returns the precedence value.
java.net.InetAddress	getRemoteAddress()	Returns the address to which media is to be transmitted.
int	getRemotePort()	Returns the port to which media is to be transmitted.
boolean	getSilenceSuppression()	Returns false if Cisco Unified Communication Manager service parameter “Silence Suppression” is set to False or True otherwise.

Related Documentation

See CiscoRTPBitRate.

CiscoRTPOutputStartedEv

The CiscoRTPOutputStartedEv event interface indicates the start of media transmission.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoRTPOutputStartedEv extends CiscoTermEv

Fields

Table 174: Fields in CiscoRTPOutputStartedEv

Interface	Field	Description
staticint	ID	None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 175: Methods in CiscoRTPOutputStartedEv

Interface	Method	Description
CiscoRTPOutputProperties	getRTPOutputProperties()	Returns the RTP output properties.
CiscoCallID	getCallID()	Returns CiscoCallID. CiscoRTPOutputStartedEv applies to CiscoCallID.getCall().
CiscoRTPHandle	getCiscoRTPHandle()	Returns a CiscoRTPHandle object. Applications can get a call reference by using CiscoProvider.getCall. If there is no call observer or there was no call observer when this event is delivered, CiscoProvider.getCall may return null.
int	getMediaConnectionMode()	Returns a CiscoMediaConnectionMode with one of the following values for mediaMode: • CiscoMediaConnectionMode.TRANSMIT_ONLY (one-way media; transmit only) • CiscoMediaConnectionMode. TRANSMIT_AND_RECEIVE (two-way media) **Note** In general, you should never get an event with mode NONE; however, if that happens, applications should ignore the event and log an error.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values, CiscoCallID, and CiscoRTPParams.

CiscoRTPOutputStoppedEv

The CiscoRTPOutputStoppedEv event indicates that the media transmission stopped.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoRTPOutputStoppedEv extends CiscoTermEv

Fields

Table 176: Fields in CiscoRTPOutputStoppedEv

Interface	Field	Description
staticint	ID	None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING,

META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 177: Methods in CiscoRTPOutputStoppedEv

Interface	Method	Description
CiscoCallID	getCallID()	Returns CiscoCallID. CiscoRTPOutputStoppedEv applies to CiscoCallID.getCall().
CiscoRTPHandle	getCiscoRTPHandle()	Returns a CiscoRTPHandle object. Applications can get a call reference by using CiscoProvider.getCall. If there is no call observer or there was no call observer when this event is delivered, CiscoProvider.getCall may return null.
int	getMediaConnectionMode()	• Returns CiscoMediaConnectionMode with one of the following values: • CiscoMediaConnectionMode. TRANSMIT_ONLY (one-way media; transmit) • onlyCiscoMediaConnectionMode. TRANSMIT_AND_RECEIVE (two-way media) **Note** In general, you should never get an event with mode NONE; however, if that happens, applications should ignore the event and log an error.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values, CiscoCallID, CiscoRTPParams, and CiscoMediaConnectionMode.

CiscoRTPPayload

The RTPPayload interface contains constants that describe RTP formats. The CiscoRTPInputProperties.getPayloadType and CiscoRTPOutputProperties.getPayloadType methods return these constants.

Interface History

Cisco Unified Communications Manager Release Number	Description
10.0(1)	Added H261, H263, H264, H264_SVC, T120, and H224 Methods.
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoRTPPayload

Fields

Table 178: Fields in CiscoRTPPayload

Interface	Fields	Description
static final int	NONSTANDARD	A nonstandard RTP payload
static final int	G711ALAW64K	G.711 64K a-law payload
static final int	G711ALAW56K	G.711 56K a-law payload
static final int	G711ULAW64K	G.711 64K u-law payload
static final int	G711ULAW56K	G.711 56K u-law payload
static final int	G722_64K	G.722 64K payload
static final int	G722_56K	G.722 56K payload
static final int	G722_48K	G.722 48K payload
static final int	G7231	G.723.1 payload
static final int	G728	G.728 payload
static final int	G729	G.729 payload
static final int	G729ANNEXA	G.729a payload
static final int	IS11172AUDIOCAP	IS11172AUDIOCAP payload

Interface	Fields	Description
static final int	IS13818AUDIOCAP	IS13818AUDIOCAP payload
static final int	ACY_G729AASSN	ACY_G729AASSN payload
static final int	DATA64	DATA64 payload
static final int	DATA56	DATA56 payload
static final int	GSM	GSM payload
static final int	ACTIVEVOICE	ACTIVEVOICE payload
static final int	WIDEBAND_256K	Wide_Band_256k payload

Methods

Table 179: Methods in CiscoRTPPayload

Interface	Method	Description
static final int	H261	The rtp payload type associated with the multimedia stream is H261.
static final int	H263	The rtp payload type associated with the multimedia stream is H263.
static final int	H264	The rtp payload type associated with the multimedia stream is H264.
static final int	H264_SVC	The rtp payload type associated with the multimedia stream is H264_SVC.
static final int	T120	The rtp payload type associated with the multimedia stream is T120.
static final int	H224	The rtp payload type associated with the multimedia stream is H224.

Related Documentation

See CiscoRTPInputProperties.getPayloadType(), CiscoRTPOutputProperties.getPayloadType(), and Constant Field Values.

CiscoRTPProperties

This interface contains the rtp properties of the multi media streams information.

Table 180: Interface History

Cisco Unified Communications Manager Release Number	Description
10.0(1)	New interface.

Declaration

public interface CiscoRTPProperties

Methods

Table 181: Methods in CiscoRTPProperties

Interface	Method	Description
InetAddress	getReceptionAddress()	Specifies the receiving IP address.
int	getReceptionPort()	Specifies the receiving port number.
InetAddress	getTransmissionAddress()	Specifies the transmitting IP address.
boolean	getTransmissionPort()	Specifies the transmitting port number.
int	getPayloadType()	Returns the payload format. The payload type can be: • CiscoRTPPayload.H261 = 100 • CiscoRTPPayload.H263_VIDEO = 101 • CiscoRTPPayload.VIDEO = 102 • CiscoRTPPayload.H264 = 103 • CiscoRTPPayload.H264_SVC = 104 • CiscoRTPPayload.T120 = 105 • CiscoRTPPayload.H224 = 106
int	getMaxBitRate()	Returns the maximum bit rate (bits per second), which is the max allowed video bit rate as negotiated by media layer. The value is the smaller of the region BW(Bandwidth) setting and BW requested by the endpoints.

CiscoSynchronousObserver

The Cisco JTAPI implementation is designed to allow applications to invoke blocking JTAPI methods such as Call.connect() and TerminalConnection.answer() from within their observer callbacks. This means that applications are not subject to the restrictions imposed by the JTAPI specification, which cautions applications against using JTAPI methods from within observer callbacks.

Normally, when an application adds a new observer to a JTAPI object, the Cisco JTAPI implementation creates an event queue and an accompanying worker thread to service the new observer. If the same observer is added to another object, its queue and thread are reused; in effect, every unique observer object has a single queue and worker thread. As noted, the advantage of this arrangement is that an application may invoke blocking JTAPI methods from within its observer callback. A subtle disadvantage, however, is that accessor methods such as Call.getConnections() and Connection.getState() may not return results that are consistent with events when invoked from within the observer callback.

For example, suppose that an application creates and connects a call from address "A" to address "B." If the application is observing address "A", it might reasonably expect that when it receives the CallActiveEv, the state of the call will be Call.ACTIVE. This is not necessarily true, because the worker thread that delivers events to the application is decoupled from the internal JTAPI thread that updates object states. In fact, if "B" rejects the call from "A, " the call object might be in either the Call.ACTIVE state or the Call.INVALID state, depending on the exact moment at which the worker thread delivers the CallActiveEv.

Many applications will not be adversely affected by this asychronous behavior. Applications that would benefit from a coherent call model during observer callbacks, however, can selectively disable the queueing logic of the Cisco JTAPI implementation. Applications that implement the CiscoSynchronousObserver interface on their observer objects declare that they want events to be delivered synchronously to its observers. Events delivered to synchronous observers will match the states of the call model objects queried from within the observer callback.

Objects that implement the CiscoSynchronousObserver interface may not invoke blocking JTAPI methods from within their event callbacks. The consequences of doing so are unpredictable, and may include deadlocking the JTAPI implementation. On the other hand, you may safely use the accessor methods of any JTAPI object, such as Call.getState() or Connection.getState(). Applications should avoid calling any interface that returns an array such as Terminal.getAddresses() in synchronous callbacks.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoSynchronousObserver

Fields

None

Methods

None

Related Documentation

None

CiscoTermActivatedEv

If a Terminal is observed and the restriction status changes to active, the system sends this event to the application.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Declaration

public interface CiscoTermActivatedEv extends CiscoProvEv

Fields

Table 182: Fields in CiscoTermActivatedEv

Interface	Field	Description
static int	ID	None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING,

META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 183: Methods in CiscoTermActivatedEv

Interface	Method	Description
javax.telephony.Terminal	getTerminal()	Returns the Terminal that is activated.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermButtonPressedEv

CiscoTermButtonPressedEv event is delivered on the TerminalObserver when a button is pressed on the Terminal. To receive this event, an application must set the filter using ciscoTermEvFilter.setButtonPressedEnabled(true). The button pressed events respond only to the numeric keypad button presses on the Terminal as listed in the constants in this interface: 0-9, *, #, A, B, C, and D.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoTermButtonPressedEv extends CiscoTermEv

Fields

Table 184: Fields in CiscoTermButtonPressedEv

Interface	Field
staticint	CHARA
staticint	CHARB
staticint	CHARC
staticint	CHARD
staticint	EIGHT
staticint	FIVE
staticint	FOUR
staticint	ID
staticint	NINE
staticint	ONE
staticint	POUND
staticint	SEVEN
staticint	SIX
staticint	STAR
staticint	THREE
staticint	TWO
staticint	ZERO

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING,

META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 185: Methods in CiscoTermButtonPressedEv

Interface	Method	Description
int	getButtonPressed()	The button pressed on the Terminal.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See CiscoTermEvFilter and Constant Field Values.

CiscoTermConnMonitoringEndEv

The system delivers the CiscoTermConnMonitoringEndEv event to the call observer when monitoring stops on the call or when call is disconnected.

Interface History

Cisco Unified Communications Manager Release	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.events.CallEv, javax.telephony.events.Ev, javax.telephony.events.TermConnEv

Declaration

public interface CiscoTermConnMonitoringEndEv extends javax.telephony.events.TermConnEv

Fields

Table 186: Fields in CiscoTermConnMonitoringEndEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 187: CiscoTermConnMonitoringEndEv Methods

Interface	Method	Description
Int	getMonitorType()	Returns the type of monitoring. The return value is always CiscoCall.SILENT_MONITOR.

Inherited Methods

From Interface javax.telephony.events.TermConnEv

getTerminalConnection

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values for more information.

CiscoTermConnMonitoringStartEv

The system delivers the CiscoTermConnMonitoringStartEv event to the call observer when monitoring starts on the call.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.events.CallEv, javax.telephony.events.Ev, javax.telephony.events.TermConnEv

Declaration

public interface CiscoTermConnMonitoringStartEv extends javax.telephony.events.TermConnEv

Fields

Table 188: Fields in CiscoTermConnMonitoringStartEv

Interface	Field
staticfinal int	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 189: CiscoTermConnMonitoringStartEv Methods

Interface	Method	Description
int	getMonitorType()	Returns the type of monitoring. The return value is always CiscoCall.Silent_Monitor.

Inherited Methods

From Interface javax.telephony.events.TermConnEv

getTerminalConnection

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values for more information.

CiscoTermConnMonitorInitiatorInfoEv

The CiscoTermConnMonitorInitiatorInfoEv event interface extends the TermConnEv interface and gets reported via the CallObserver on the monitor target (agent). This interface gives information about the monitor initiator (supervisor) when a monitor session gets established.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.
8.0(1)	A new API getTransactionID() will be exposed to retrieve the transaction ID.

Superinterfaces

javax.telephony.events.CallEv, javax.telephony.events.Ev, javax.telephony.events.TermConnEv

Declaration

public interface CiscoTermConnMonitorInitiatorInfoEv extends javax.telephony.events.TermConnEv

Fields

Table 190: Fields in CiscoTermConnMonitorInitiatorInfoEv

Interface	Fields
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 191: Methods in CiscoTermConnMonitorInitiatorInfoEv

Interface	Method	Description
CiscoMonitorInitiatorInfo	getCiscoMonitorInitiatorInfo()	Returns the Terminal name and Address of the monitor initiator.
int	getTransactionID()	Returns the transaction ID.

Inherited Methods

From Interface javax.telephony.events.TermConnEv

getTerminalConnection

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermConnMonitorTargetInfoEv

The CiscoTermConnMonitorTargetInfoEv event interface extends the TermConnEv interface and gets reported via the CallObserver on monitor initiator. This interface provides information about the monitor target (agent) when a monitor session gets established.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.
8.0(1)	A new getTransactionID() will be exposed to retrieve the transaction ID.

Superinterfaces

javax.telephony.events.CallEv, javax.telephony.events.Ev, javax.telephony.events.TermConnEv

Declaration

public interface CiscoTermConnMonitorTargetInfoEv extends javax.telephony.events.TermConnEv

Fields

Table 192: Fields in CiscoTermConnMonitorTargetInfoEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 193: Methods in CiscoTermConnMonitorTargetInfoEv

Interface	Method	Description
CiscoMonitorTargetInfo	getCiscoMonitorTargetInfo()	Returns the Terminal name and Address of the monitor target.
int	getTransactionID()	Returns the transaction ID.

Inherited Methods

From Interface javax.telephony.events.TermConnEv

getTerminalConnection

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermConnPrivacyChangedEv

The system sends the CiscoTermConnPrivacyChangedEv event when the privacy status of a TeminalConnection changes. If privacy is active, the user cannot Barge into the Call.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoTermConnPrivacyChangedEv

Fields

Table 194: Fields in CiscoTermConnPrivacyChangedEv

Interface	Field
staticint	ID

Methods

Table 195: Methods in CiscoTermConnPrivacyChangedEv

Interface	Method	Description
javax.telephony. TerminalConnection	getTerminalConnection()	Returns the TerminalConnection where privacy changed. You can call getPrivacyStatus on the TerminalConnection to check its privacy status.

Related Documentation

See Constant Field Values and CiscoTerminalConnection.getPrivacyStatus().

CiscoTermConnRecordingEndEv

The JTAPI delivers CiscoTermConnRecordingEndEv to the call observer when call recording stops.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.events.CallEv, javax.telephony.events.Ev, javax.telephony.events.TermConnEv

Declaration

public interface CiscoTermConnRecordingEndEv extends javax.telephony.events.TermConnEv

Fields

staticintID

Inherited Fields

Fields inherited from interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.events.TermConnEv

getTerminalConnection

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermConnRecordingStartEv

The JTAPI delivers CiscoTermConnRecordingStartEv to the call observer when call recording starts.

Superinterfaces

javax.telephony.events.CallEv, javax.telephony.events.Ev, javax.telephony.events.TermConnEv

Declaration

public interface CiscoTermConnRecordingStartEv extends javax.telephony.events.TermConnEv

Fields

Table 196: Fields in CiscoTermConnRecordingStartEv

Interface	Field
static int	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.events.TermConnEv

getTerminalConnection

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermConnRecordingTargetInfoEv

The JTAPI delivers CiscoTermConnRecordingTargetInfoEv to the call observer of the recording initiator.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.events.CallEv, javax.telephony.events.Ev, javax.telephony.events.TermConnEv

Declaration

public interface CiscoTermConnRecordingTargetInfoEv extends javax.telephony.events.TermConnEv

Fields

Table 197: Fields in CiscoTermConnRecordingTargetInfoEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN,

META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 198: Methods in CiscoTermConnRecordingTargetInfoEv

Interface	Method	Description
CiscoRecorderInfo	getCiscoRecorderInfo()	Returns CiscoRecorderInfo, which provides the terminal name and address of the recording device.

From Interface javax.telephony.events.TermConnEv

getTerminalConnection

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values and CiscoRecorderInfo.

CiscoTermConnRecordingFailedEv

The JTAPI delivers CiscoTermConnRecordingFailedEv to the call observer when call recording failed.

Superinterfaces

javax.telephony.events.CallEv, javax.telephony.events.Ev, javax.telephony.events.TermConnEv

Declaration

public interface CiscoTermConnRecordingFailedEv extends TermConnEv

Fields

Table 199: Fields in CiscoTermConnRecordingStartEv

Interface	Field
static int	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.events.TermConnEv

getTerminalConnection

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermConnSelectChangedEv

The JTAPI sends CiscoTermConnSelectChangedEv when the call select status of a TerminalConnection changes, either by feature invocation or manually.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.events.CallEv, javax.telephony.events.Ev, javax.telephony.events.TermConnEv

Declaration

public interface CiscoTermConnSelectChangedEv extends javax.telephony.events.TermConnEv

Fields

Table 200: Fields in CiscoTermConnSelectChangedEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.events.TermConnEv

getTerminalConnection

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermCreatedEv

The JTAPI sends the CiscoTermCreatedEv event to the provider observer of the application when a CiscoTerminal gets added to the provider domain.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Declaration

public interface CiscoTermCreatedEv extends CiscoProvEv

Fields

Table 201: Fields in CiscoTermEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 202: Methods in CiscoTermCreatedEv

Interface	Method	Description
javax.telephony.Terminal	getTerminal()	Returns the Terminal object for which this event was sent.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermDataEv

The JTAPI sends the CiscoTermDataEv event to the terminal observer when the phone receives XSI data (XML object).

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoTermDataEv extends CiscoTermEv

Fields

Table 203: Fields in CiscoTermDataEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 204: Methods in CiscoTermDataEv

Interface	Method	Description
java.lang.String	getData()	Deprecated.Use byte[] getTermData
byte[]	getTermData()	Returns an XML-encoded byte array corresponding to the XSI data that the phone received.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermDeviceStateActiveEv

The CiscoTermDeviceStateActiveEv event gets sent to the Terminal Observer if any of the addresses on the terminal have an outgoing call in any state or an incoming call with TerminalConnection and CallCtlTerminalConnection in ACTIVE and TALKING state respectively.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoTermDeviceStateActiveEv extends CiscoTermEv

Fields

Table 205: Fields in CiscoTermDeviceStateActiveEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermDeviceStateAlertingEv

The CiscoTermDeviceStateAlertingEv event gets sent to the Terminal Observer if none of the Addresses on the Terminal have an outgoing call or an incoming call with Connection in CallCtlConnection.Established

state, and at least one of the addresses on the Terminal has an incoming Call with Connection in CallCtlConnection.OFFERED or CallCtlConnection.ALERTING State.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoTermDeviceStateAlertingEv extends CiscoTermEv

Fields

Table 206: Fields in CiscoTermDeviceStateAlertingEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermDeviceStateHeldEv

The CiscoTermDeviceStateHeldEv event gets sent to the Terminal Observer if all of the calls on the addresses of the Terminal have TerminalConnection in CallCtlTerminalConnection.HELD state.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoTermDeviceStateHeldEv extends CiscoTermEv

Fields

Table 207: Fields in CiscoTermDeviceStateHeldEv

Interface	Field
staticint	ID

Inherited Fields

Fields Inherited From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermDeviceStateIdleEv

The CiscoTermDeviceStateIdleEv event gets sent to the Terminal Observer if there are no calls on any of the Addresses of the Terminal.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoTermDeviceStateIdleEv extends CiscoTermEv

Fields

Table 208: Fields in CiscoTermDeviceStateIdleEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermDeviceStateWhisperEv

The CiscoTermDeviceStateActiveEv event gets sent to Terminal Observer if atleast one of the Addresses on the Terminal is an intercom target and has an intercom call with the TerminalConnection/CallCtlTerminalConneciton in State Passive/Bridged state. In this state, the user can initiate new outgoing calls and receive new incoming calls.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoTermDeviceStateWhisperEv extends CiscoTermEv

Fields

Table 209: Fields in CiscoTermDeviceStateWhisperEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermDNDOptionChangedEv

The CiscoTermDNDOptionChangedEv event is delivered to the terminal observer if DND option is changed. This event is delivered only if the filter to receive events is enabled by the application. This event is provided on application observer

History

Cisco Unified Communications Manager Release	Description
7.0(1)	Added the extension.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

public interface CiscoTermDNDOptionChangedEv

extends CiscoTermEv

Fields

Table 210: CiscoTermDNDOptionChangedEv Fields

Interface	Field
static final int	ID

Table 211: Inherited Fields

From Interface javax.telephony.events.Ev
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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Table 212: Inherited Fields

From Interface javax.telephony.events.Ev
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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 213: CiscoTermDNDOptionChangedEv Methods

Interface	Method	Description
Int	getDNDOption()	This interface returns the current DND option to the application. It returns int dndOption.

Table 214: Inherited Methods

From Interface javax.telephony.events.Ev
getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Table 215: Inherited Methods

From Interface javax.telephony.events.TermEv
getTerminal

Table 216: Inherited Methods

From Interface javax.telephony.events.Ev
getCause, getID, getMetaCode, getObserved, isNewMetaEvent

See also Constant Field Values. and CiscoTermEv.

CiscoTermDNDStatusChangedEv

The CiscoTermDNDStatusChangedEv event gets delivered to the Terminal Observer if the DND status changes, provided that the application has enabled the filter to receive events.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoTermDNDStatusChangedEv extends CiscoTermEv

Fields

Table 217: Fields in CiscoTermDNDStatusChangedEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 218: Methods in CiscoTermDNDStatusChangedEv

Interface	Method	Description
boolean	getDNDStatus()	Returns the current DND status to the application.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See CiscoTermEvFilter, CiscoTermEv, and Constant Field Values.

CiscoTermEv

The CiscoTermEv interface, which extends the JTAPI core javax.telephony.events.TermEv interface, serves as the base interface for all Cisco-extended JTAPI Terminal events. Every Call-related event in this package extends this interface, directly or indirectly.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Subinterfaces

CiscoMediaOpenLogicalChannelEv, CiscoRTPInputKeyEv, CiscoRTPInputStartedEv, CiscoRTPInputStoppedEv, CiscoRTPOutputKeyEv, CiscoRTPOutputStartedEv, CiscoRTPOutputStoppedEv,

CiscoTermButtonPressedEv, CiscoTermDataEv, CiscoTermDeviceStateActiveEv, CiscoTermDeviceStateAlertingEv, CiscoTermDeviceStateHeldEv, CiscoTermDeviceStateIdleEv, CiscoTermDeviceStateWhisperEv, CiscoTermDNDOptionChangedEv, CiscoTermDNDStatusChangedEv, CiscoTermInServiceEv, CiscoTermOutOfServiceEv, CiscoTermRegistrationFailedEv, CiscoTermSnapshotCompletedEv, CiscoTermSnapshotEv

Declaration

public interface CiscoTermEv extends CiscoEv, javax.telephony.events.TermEv

Fields

None

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See CallEv.

CiscoTermEvFilter

An application can use the CiscoTermEvFilter interface to selectively restrict those Terminal events that are not of interest.

Interface History

Cisco Unified Communications Manager Release Number	Description
11.5(1)	Added getHuntLogStatusChangedEvFilter() and setHuntLogStatusChangedEvFilter(boolean filterValue) methods.
10.0(1)	Added getMultiMediaStreamsInfoEvFilter() and setMultiMediaStreamsInfoEvFilter(boolean filterValue) methods.
7.1(1 and 2)	Created history table to track changes.

Declaration

public interface CiscoTermEvFilter

Fields

None

Methods

Table 219: Methods in CiscoTermEvFilter

Interface	Method	Description
boolean	getDeviceDataEnabled()	Returns the event filter status of the CiscoTermDataEv event for the Terminal. The default value is disabled. Returns True if the event filter is enabled, or false if the event filter is disabled.
void	setDeviceDataEnabled(booleanenabled)	Enables or disables the CiscoTermDataEv events for the Terminal.

Interface	Method	Description
boolean	getButtonPressedEnabled()	Returns the event filter status of the CiscoTermButtonPressedEv event for the Terminal. The default value is disabled. Returns:True if the event filter is enabled, or false if the event filter is disabled.
void	setButtonPressedEnabled(booleanenabled)	Enables or disables CiscoTermButtonPressedEv events for the Terminal.
boolean	getRTPEventsEnabled()	Returns the event filter status of the CiscoRTPInputStartedEv, CiscoRTPOutputStartedEv, CiscoRTPInputStoppedEv, and CiscoRTPOutputStoppedEv events for the Terminal. The Default value is disabled. Returns:True if the event filter is enabled, or false if the event filter is disabled.
void	setRTPEventsEnabled(booleanenabled)	Enables or disables CiscoRTPInputStartedEv, CiscoRTPOutputStartedEv, CiscoRTPInputStoppedEv and CiscoRTPOutputStoppedEv events for the Terminal.
boolean	getSnapshotEnabled()	Returns the event filter status of the CiscoTermSnapshotEv and CiscoTermSnapshotCompletedEv events for the Terminal. If disabled, neither event gets sent to applications. Returns:True if the event filter is enabled, or false if the event filter is disabled.
void	setSnapshotEnabled(booleanenabled)	Enable or disables CiscoTermSnapshotEv and CiscoTermSnapshotCompletedEv for the Terminal.
boolean	getRTPKeyEventsEnabled()	Returns the event filter status of the CiscoRTPInputKeyEv and CiscoRTPOutputKeyEv events for the Terminal. Returns:True if the event filter is enabled, or false if the event filter is disabled.
void	setRTPKeyEventsEnabled(booleanenabled)	Enables or disables the CiscoRTPInputKeyEv and CiscoRTPOutputKeyEv events for the Terminal.
boolean	getDeviceStateActiveEvFilter()	Returns the event filter status of the CiscoTermDeviceStateActiveEv event for the Terminal. Returns:True if the event filter is enabled, or false if the event filter is disabled.
boolean	getDeviceStateHeldEvFilter()	Returns the event filter status of the CiscoTermDeviceStateHeldEv event for the Terminal. Returns: True if the event filter is enabled, or false if the event filter is disabled.
boolean	getDeviceStateAlertingEvFilter()	Returns the event filter status of the CiscoTermDeviceStateAlerting event for the Terminal. Returns:True if the event filter is enabled, or false if the event filter is disabled.

Interface	Method	Description
boolean	getDeviceStateIdleEvFilter()	Returns the CiscoTermDeviceStateIdleEv filter status. Returns:True if the event filter is enabled, or false if the event filter is disabled.
void	setDeviceStateActiveEvFilter(booleanfilterValue)	Enables or disables the CiscoTermDeviceStateActiveEv filter for the Terminal. Default value is disable.
void	setDeviceStateHeldEvFilter(booleanfilterValue)	Enables or disables the CiscoTermDeviceStateHeldEv filter for the Terminal. Default value is disable
void	setDeviceStateAlertingEvFilter(booleanfilterValue)	Enables or disables the CiscoTermDeviceStateAlertingEv filter for the Terminal. Default value is disable
void	setDeviceStateIdleEvFilter(booleanfilterValue)	Enables or disables the CiscoTermDeviceStateIdleEv filter for the Terminal. Default value is disable
boolean	getDeviceStateWhisperEvFilter()	Returns the CiscoTermDeviceStateWhisperEv filter status on the Terminal.
boolean	getDNDChangedEvFilter()	Returns the CiscoTermDNDStatusChangedEv filter status Returns:the CiscoTermDNDStatusChangedEv Filter status on the Terminal.
void	setDNDChangedEvFilter(booleanfilterValue)	Enables or disables the CiscoTermDNDStatusChangedEv filter for the Terminal. Parameters:filterValue - void setDeviceStateWhisperEvFilter(booleanfilterValue) Enables or disables the CiscoTermDeviceStateWhisperEv filter for the Terminal. The default value is disable.
boolean	getDNDOptionChangedEvFilter()	This interface can be used to get CiscoTermDNDOptionChangedEv filter status . Returns:the CiscoTermDNDOptionChangedEv Filter status on the Terminal.
void	setDNDOptionChangedEvFilter(booleanfilterValue)	This interface is provided for enabling/disabling the CiscoTermDNDOptionChangedEv filter for the Terminal Parameters:filterValue.
boolean	getMultiMediaStreamsInfoEvFilter()	This interface can be used to get CiscoMultiMediaStreamsInfoEv filter status.
void	setMultiMediaStreamsInfoEvFilter(boolean filterValue)	This interface is provided for enabling/disabling the CiscoMultiMediaStreamsInfoEv filter for the Terminal.
boolean	getHuntLogStatusChangedEvFilter()	This method is used get the value of filter, the value of filter is false by default.

Interface	Method	Description
void	setHuntLogStatusChangedEvFilter(boolean filterValue)	This method is used to set the filter, if filter value is true, the event CiscoTermHuntLogStatusChangedEv is received by the application when the value of huntLogStatus is changed.

Related Documentation

None

CiscoTerminal

Standard JTAPI does not support the notion of dynamic terminal registration. The CiscoTerminal interface extends the standard terminal interface to do so. All Cisco Unified Communications Manager devices are represented by CiscoTerminals, and you can query all CiscoTerminals to determine whether they are currently IN_SERVICE or OUT_OF_SERVICE.

If the Cisco Unified Communications Manager device represented by the CiscoTerminal is an IP telephone, for instance, it goes OUT_OF_SERVICE if it loses its network connection. Other types of devices, such as CiscoMediaTerminal, get registered on demand by applications, and may be IN_SERVICE or OUT_OF_SERVICE accordingly.

CiscoTerminal includes an API getIPAddressingMode(). This interface returns the configured IP Addressing Mode of the CiscoTerminal.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.
7.1.(3)	This interface is enhanced to: • Get the IP addresses of the terminal • Get the outbound call roll over configuration of the terminal. New interfaces expose consult call roll over, out bound call rollover, Join across lines (JAL) and Direct Transfer Across Lines (DTAL) capability of the terminal. A terminal can have different capability when feature is invoked on the phone and when feature is invoked by application. It does not indicate if the entire configuration required for the feature is enabled (for example, Join softkey must be configured in the template for the feature). • Get registered state of the terminal

Cisco Unified Communications Manager Release Number	Description
8.0(1)	Added the following new APIs: • APIs pickup(Address pickingAddress) groupPickup(Address pickingAddress • String pickupGroupNumber) • directedPickup(Address pickingAddress, String pickupGroupNumber), • Pickup(Address pickingAddress) for picking up calls from different flavors of Call Pickup Groups. • API getLoginType () which returns the LoginType on the terminal.
8.5(1)	A new API, isBuiltInBridgeEnabled(). is added.
10.0(1)	A new interface, public interface CiscoTerminal, is added on CiscoTerminal to determine the multimedia capabilities of the terminal.
11.5(1)	Added getHuntLogStatus() throwsInvalidStateException and setHuntLogStatus(int huntLogStatus)throws InvalidStateException,MethodNotSupportedException, methods.

Sample Code

```
public class MyTerminalObserver implements TerminalObserver
{
  public void terminalChangedEvent (TermEv[] evlist) {
    for(int i = 0; evlist ! =  null && i < evlist.length; i++)
    {
      …
      if ( evlist[i] instance of TermInServiceEv)
      {
        CiscoTerminal term  =  (CiscoTerminal)  (((CiscoTermEv)evlist[i]
).getTerminal());
        if(!term instanceof CiscoMediaTerminal && ! term instanmceof
CiscoRouteterminal)
        {
          try
          {
            if ( term. isBuiltInBridgeEnabled())
            {
              System.out.println("Build in Bridge is enabled for terminal" +
                tern.getName());
              else
              {
                System.out.println("Build in Bridge is disabled for terminal" +
                  tern.getName());
              }
            }
            catch(Exception)
            {
              System.out.println("Exception caught");
            }
          }
```

```
            }
        }
    }
}
```

Superinterfaces

CiscoObjectContainer, javax.telephony.Terminal

Subinterfaces

CiscoMediaTerminal, CiscoRouteTerminal

Declaration

public interface CiscoTerminal extends javax.telephony.Terminal, CiscoObjectContainer

Fields

Table 220: Fields in CiscoTerminal

Interface	Field	Description
static final int	DEVICE_HUNT_LOGGED_IN	This constant depicts that the terminal is logged into the hunt group.
static final int	DEVICE_HUNT_LOGGED_OUT	This constant depicts that the terminal is logged out of the hunt group.
static final int	DEVICE_HUNT_NOT_APPLICABLE	This constant depicts that the terminal does not have the capability either to log in or log out of the hunt group.
static final int	OUT_OF_SERVICE	This Constant Field Values returned by the getState() interface on CiscoTerminal indicates that the CiscoTerminal is out of service.
static final int	IN_SERVICE	This constant value returned by the getState() interface on CiscoTerminal indicates that the CiscoTerminal is in service.
static final int	DEVICESTATE_IDLE	This constant value returned by the getDeviceState() interface on CiscoTerminal indicates that the CiscoTerminal currently has no calls on any Addresses.
static final int	DEVICESTATE_ACTIVE	This constant value returned by the getDeviceState() interface on CiscoTerminal indicates that the CiscoTerminal has at least one active call on an Address.

Interface	Field	Description
static final int	DEVICESTATE_ALERTING	This constant value returned by the getDeviceState() interface on CiscoTerminal indicates that the CiscoTerminal has at least one alerting call, but no active call, on an Address.
static final int	DEVICESTATE_HELD	This constant value returned by the getDeviceState() interface on CiscoTerminal indicates that the CiscoTerminal has at least one held call, but no alerting or active calls, on an Address.
static final int	DEVICESTATE_UNKNOWN	This constant value returned by the getDeviceState() interface on CiscoTerminal indicates that the CiscoTerminal DeviceState is Unknown. This state may get returned if the device state filters are disabled.
static final int	DEVICESTATE_WHISPER	This constant value returned by the getDeviceState() interface on CiscoTerminal indicates that the CiscoTerminal has at least one intercom call with one-way media, but has no held, alerting, or active calls on an Address.
static final int	UNKNOWN_ENCODING	Indicates that the CiscoTerminal.getSupportedEncoding () for this terminal is UNKNOWN.
static final int	NOT_APPLICABLE	Indicates that the CiscoTerminal.getSupportedEncoding () for this CiscoMediaTerminal or RoutePoint is NOT_APPLICABLE.
static final int	ASCII_ENCODING	Indicates that the CiscoTerminal.getSupportedEncoding () for this terminal is ASCII and that this terminal supports only ASCII_ENCODING.
static final int	UCS2UNICODE_ENCODING	Indicates that the CiscoTerminal.getSupportedEncoding () for this terminal is UCS2UNICODE_ENCODING.
static final int	DND_OPTION_NONE	This constant value returned by the getDNDOption() interface on CiscoTerminal indicates that the DND option configured is None.
static final int	DND_OPTION_RINGER_OFF	This constant value returned by the getDNDOption() interface on CiscoTerminal indicates that the DND option configured is Ringer Off. If DND is enabled on the phone, the phone would not ring if a call lands there.

Interface	Field	Description
static final int	DND_OPTION_CALL_REJECT	This constant value returned by the getDNDOptions() interface on CiscoTerminal indicates that the DND option configured is Call Reject. If DND is enabled on the phone, all calls to the phone will get rejected, except for shared lines.
static final int	IP_ADDRESSING_MODE_UNKNOWN	This indicates that IPAddressing mode is Unknown.
static final int	IP_ADDRESSING_MODE_IPV4	This indicates that IPAddressing mode is IPv4
static final int	IP_ADDRESSING_MODE_IPV6	This indicates that IPAddressing mode is IPv6
static final int	IP_ADDRESSING_MODE_IPV4_V6	This indicates that IPAddressing mode is both IPv4 and IPv6
static final int	IP_ADDRESSING_MODE_UNKNOWN_ANATRED	This is reserved IP Addressing constant for ANAT in Cisco Unified Communications Manager. It is not used in JTAPI

Methods

Table 221: Methods in CiscoTerminal

Inteface	Method	Description
int	getLoginType()	This method returns the value NO_EM_LOGIN, NATIVE_LOGIN or VISITOR_LOGIN to indicate whether the terminal is local to the cluster or not when the EM login is done.
Static final int	CiscoTerminal.NO_LOGIN	This indicates that there has been no EM login done into the terminal. It will have an integer value of 0.
	CiscoTerminal.NATIVE_LOGIN	This indicates that the terminal is part of the local cluster when an EM login is done into it with a profile that belongs to the same cluster. It will have an inteher value of 1.
	CiscoTerminal.VISITOR_LOGIN	This indicates that the terminal is part of the visiting cluster when an EM login is done into it with a profile that is not local to the cluster. It will have an integer value of 2.
int	getRegistrationState()	Deprecated This method has been replaced by the getState() method. Returns the state of this terminal. The state may be any of the following constants: • CiscoTerminal.OUT_OF_SERVICE • CiscoTerminal.IN_SERVICE

Inteface	Method	Description
int	getState()	Returns the state of this terminal. The state may be any of the following constants: • CiscoTerminal.OUT_OF_SERVICE • CiscoTerminal.IN_SERVICE
CiscoRTPInputProperties	getRTPInputProperties()	Returns the properties to be used for the RTP input stream associated with the ACTIVE TerminalConnection on this terminal. The CiscoTerminal must be in the CiscoTerminal.REGISTERED state, its Provider must be in the Provider.IN_SERVICE state, and Terminal.getTerminalConnections () must return at least one terminal connection in the TerminalConnection.ACTIVE state. **Throws** javax.telephony.InvalidStateException
CiscoRTPOutputProperties	getRTPOutputProperties()	Returns the properties to be used for the RTP output stream associated with the ACTIVE TerminalConnection on this terminal. The CiscoTerminal must be in the CiscoTerminal.REGISTERED state, its Provider must be in the Provider.IN_SERVICE state, and Terminal.getTerminalConnections () must return at least one terminal connection in the TerminalConnection.ACTIVE state. **Throws** javax.telephony.InvalidStateException
java.lang.String	sendData(java.lang. StringterminalData)	Deprecated Use CiscoTerminal.sendData (byte[]). **Throws** javax.telephony.InvalidStateException javax.telephony.MethodNotSupportedException
byte[]	sendData(byte[]terminalData)	The CiscoTerminal must be in the CiscoTerminal.IN_SERVICE state, and its Provider must be in the Provider.IN_SERVICE state. Applications can push the XSI object in the byte format to the phone with this interface. If the phone receives the data, this method returns successfully. Applications may only send 2000 bytes of data with this interface. Requests carrying excess data get rejected. **Throws** PlatformException (The data did not get sent successfully), javax.telephony.InvalidStateException, and javax.telephony.MethodNotSupportedException
CiscoTermEvFilter	getFilter()	Retrieves the filter object associated with the terminal.

Inteface	Method	Description
void	setFilter(CiscoTermEvFilter terminalEvFilter)	Filters the events that get delivered to the TerminalObserver. You can call this method at any time, but the typical usage is to set up the terminal events as part of initialization or when a CiscTermCreatedEv indicates that the system created a new terminal. • Example 1—One use might be to turn on the button-pressed events that normally do not get not delivered. Terminal term = provider.getTerminal (name); if (term instanceof CiscoTerm) { CiscoTerm ciscoTerm = (CiscoTerm)term; CiscoTermEvFilter filter = ciscoTerm.getFilter (); filter.setButtonPressedEnabled (true); } term.addObserver (terminalObserver) • Example 2—Another use might be turning off events that are not of interest to an application. For example, an application doing pure call control could turn off the media (RTP) events as follows: Terminal term = provider.getTerminal (name); if (term instanceof CiscoTerm) { CiscoTerm ciscoTerm = (CiscoTerm)term; CiscoTermEvFilter filter = ciscoTerm.getFilter (); filter.setRTPEventsEnabled (false); ciscoTerm.setFilter (filter); } term.addObserver (terminalObserver); term.getAddresses () [0].addCallObserver (callObserver) **Note** Adding a CallObserver (without explicitly setting a filter) turns the RTP events on. This behavior of Cisco JTAPI Release 1.4 and earlier is still preserved. If an explicit setFilter call gets made, the filter settings will take effect. The RTP events will not get delivered for the previous code snippet, but will get delivered for the following example: Terminal term = provider.getTerminal (name); term.addObserver (terminalObserver); term.getAddresses () [0].addCallObserver (callObserver).

Inteface	Method	Description
javax.telephony.TerminalConnection	unPark(javax.telephony.Address UnParkAddress, java.lang.StringParkedAt)	Returns a terminalConnection. The CiscoTerminal must be in the CiscoTerminal.IN_SERVICE state and its Provider must be in the Provider.IN_SERVICE state. This method takes an address and string as input. **Parameters** • UnParkAddress - Any address on the terminal • ParkedAt - A string identifying is system park port where a call was previously parked. The system returns this string when the call gets parked. **Throws** • javax.telephony.InvalidStateException (The CiscoTerminal.getState() is not IN_SERVICE) • PlatformException - Any other error occurred while unparking (for example, the Unpark number is not valid). • javax.telephony.InvalidArgumentException • javax.telephony.ResourceUnavailableException
int	getDeviceState()	• Returns the DeviceState of this terminal. The DeviceState is the accumulative call state of all the addresses on the terminal. The state may be any of the following constants: • CiscoTerminal.DEVICESTATE_ILDE • CiscoTerminal.DEVICESTATE_ACTIVE • CiscoTerminal.DEVICESTATE_ALERTING • CiscoTerminal.DEVICESTATE_HELD • CiscoTerminal.DEVICESTATE_UNKNOWN • CiscoTerminal.DEVICESTATE_WHISPER **Throws** • javax.telephony.InvalidStateException —CiscoTerminal.getState() is not IN_SERVICE.

Inteface	Method	Description
int	getSupportedEncoding()	Returns the supported encoding types for this terminal. Use this method to check whether a terminal supports Unicode. To access this information, the terminal must be in the CiscoTerminal.IN_SERVICE state. The supportedEncoding is one of the following constants: • CiscoTerminal.UNKNOWN_ENCODING • CiscoTerminal.ASCII_ENCODING • CiscoTerminal.UCS2UNICODE_ENCODING • CiscoTerminal.NOT_APPLICABLE **Throws** • javax.telephony.InvalidStateException
int	getLocale()	Returns the locale that this terminal supports. To access this method, the terminal must be in the CiscoTerminal.IN_SERVICE state. **Throws** • javax.telephony.InvalidStateException —CiscoTerminal.getState() is not IN_SERVICE.
boolean	isRestricted()	Returns the restriction status of this terminal. If a terminal is restricted, all associated addresses on the terminal are also restricted. Returns:True if terminal is restricted; otherwise false.
void	createSnapshot()	Generates a CiscoTermSnapshotEv event, which contains the security status of the current active call on the terminal. To access this method, the terminal must be in the CiscoTerminal.IN_SERVICE state and CiscoTermEvFilter.setSnapshotEnabled () must be set to true. **Throws** • javax.telephony.InvalidStateException —CiscoTerminal.getState() is not IN_SERVICE.
java.lang.String	getAltScript()	Returns the locale alternate script that this terminal supports. An empty return value indicates that this terminal does not support or is not configured with an alternate script. To access this method, the terminal must be in the CiscoTerminal.IN_SERVICE state. **Throws** • javax.telephony.InvalidStateException —CiscoTerminal.getState() is not IN_SERVICE.

Inteface	Method	Description
int	getProtocol()	Reports the terminal protocol (SCCP, SIP, or none) and returns the protocol of this terminal as one of the following constants: • CiscoTerminalProtocol.PROTOCOL_NONE • CiscoTerminalProtocol.PROTOCOL_SCCP • CiscoTerminalProtocol.PROTOCOL_SIP
void	setDNDStatus(booleandndStatus)	Sets the DND status, which enables or disables the DND feature. This feature does not apply to route points. **Parameters** • dndStatus **Throws** • javax.telephony.InvalidStateException —CiscoTerminal.getState() is not IN_SERVICE.
boolean	getDNDStatus()	Reports the DND status and returns dndStatus. **Throws** • javax.telephony.InvalidStateException —CiscoTerminal.getState() is not IN_SERVICE.
int	getDNDOption()	Returns the value of the DND option. This value is not significant for a CiscoMediaTerminal or CiscoRouteTerminal because the DND feature applies only to physical phones. The DND option can be any of the following constants: • CiscoTerminal.DND_OPTION_NONE • CiscoTerminal.DND_OPTION_RINGER_OFF • CiscoTerminal.DND_OPTION_CALL_REJECT **Throws** • javax.telephony.InvalidStateException —CiscoTerminal.getState() is not IN_SERVICE.
java.lang.String	getEMLoginUsername()	Returns extension mobility (EM) login user name. If no EM user has logged into Terminal, this interface will return null/empty string **Throws** • javax.telephony.InvalidStateException —if CiscoTerminal.getState() is not IN_SERVICE. **Note** You must use this API with CiscoTerminal.getLoginType() to determine if an EM login is done.

Inteface	Method	Description
InetAddress	getIPV4Address()	Returns IPV4 ip address of the terminal **Throws** • InvalidStateException • MethodNotSupportedException
InetAddress	getIPV6Address()	Returns IPV6 ip address of the terminal **Throws** • InvalidStateException • MethodNotSupportedException
Call	pickup (Address pickingAddress)	This method picks up the longest ringing call from the Pickup Group to which pickingAddress belongs to. pickingAddress is the Address on the Terminal where the Call is picked up. **Parameters** pickingAddress that specifies which Address the Terminal object should do the pickup on.
Call	groupPickup(Address pickingAddress, String pickupGroupNumber)	This method picks up the longest ringing call from the specified pickupGroupNumber at the pickingAddress. **Parameters** • pickingAddress that specifies which Address the Terminal object should do the pickup on. • Additional String object that respresents the number of the pickup group you wish to answer a call from, as set in the CUCM.
Call	directedPickup(Address pickingAddress, String ringingDN)	This method picks up the call from the specified ringingDN at the pickingAddress. The ringingDN must be in the same Pickup Group as the pickingAddress. **Parameters** • pickingAddress that specifies which Address the Terminal object should do the pickup on. • Additional String object that respresents the specific DN the application wishes to pick up for a directed call pickup request.

Inteface	Method	Description
Call	otherPickup(Address pickingAddress)	This method picks up a call based upon the priority of the associated pickupgroups. Within the group, if there are more than one call, the longest ringing call will be picked up. pickingAddress is the Address on the Terminal where the Call is picked up. **Parameters** pickingAddress that specifies which Address the Terminal object should do the pickup on.
int	getRollOverConfig()	Returns the roll over configuration of the terminal. **Throws** • InvalidStateException.
public final static int	NO_ROLLOVER	This indicates that the terminal is configured with no roll over.
public final static int	ROLLOVER_ANY_DN	This indicates that calls can roll over to any address on the terminal.
public final static int	ROLLOVER_SAME_DN	This indicates that calls can roll over to address that match the name(DN).
public static final int	PHONE_USER	Application can use this to find the capability of the terminal when used manually by user.
publis static final int	APPLICATION	Application can use to this to find the capability of the terminal to invoke the feature from application.
boolean	canConsultCallRollOver(int which_user)	Which user can be CiscoTerminal. PHONE_USER or CiscoTerminal.APPLICATION **Throws** • InvalidStateException.
boolean	canOutBoundCallRollOver(int which_user)	• InvalidStateException.
boolean	canJoinAcrossLines(int which_user)	This interface returns True if calls on different addresses of this terminal can be conferenced. This interface returns True for terminals that support Connected Transfer or Conference Across Lines. **Throws** • InvalidStateException.

Inteface	Method	Description
boolean	canDirectTransferAcrossLines(int which_user)	This interface returns True if calls on different addresses of this terminal can be transferred. This interface returns True for terminals that support Connected Transfer or Conference Across Lines. **Throws** • InvalidStateException.
boolean	canJoinOnSameLine (int which_user)	**Throws** • InvalidStateException.
boolean	canDirectTransferOnSameLine(int which_user)	**Throws** • InvalidStateException.
boolean	isRegistered()	Returns true if terminal is registered with Cisco Unified Communications Manager else false.
public boolean	isBuiltInBridgeEnabled()	Returns true if Built-In-Bridge is enabled on the terminal, else returns false.
void	register()	This method registers Cisco Unified Client Services Framework device to Extend mode, which will be represented as a CiscoRemoteTerminal. This is intended to be used by application monitoring Cisco Unified Client Services Framework device to enable its Extend mode from its softphone (SIP)/deskphone mode. The successful effect of this method is to register the device and present as a CiscoRemoteTerminal terminal type (if it is not already a CiscoRemoteTerminal), and be able to use Cisco Extend & Connect (CTI Remote Device) supported features with remote destinations. In any case when it switches in between Softphone/Deskphone & Extend modes that results in a terminal switching (e.g. CiscoTerminal->CiscoRemoteTerminal), there will be provider events sent to application: CiscoAddrRemovedEv, CiscoTermRemovedEv, CiscoAddrCreatedEv, CiscoTermCreatedEv. Any observers on the terminal and address will be removed, application needs to de-reference the old terminal object and re-add any desired observers. Note that CiscoProvider must be in IN_SERVICE state, otherwise CiscoRegistrationException about InvalidStateException will be thrown; or if terminal is already registered by this application in Extend mode or the registration fails for any reason, CiscoRegistrationException will be thrown. And if terminal type is not CiscoTerminal or CiscoRemoteTerminal and if terminal is not a Cisco Unified Client Services Framework device, then MethodNotSupportedException will be thrown.

Inteface	Method	Description
void	unregister()	This method unregisters Cisco Unified Client Services Framework device from Extend mode. Its terminal type will remain as CiscoRemoteTerminal, and application can explicitly register to CUCM to switch back to its softphone/deskphone mode. This is intended to be used by application monitoring Cisco Unified Client Services Framework device in Cisco Extend Connect mode to disable its Cisco Extend Connect mode. The successful effect of this method is to unregister the device but retains as a CiscoRemoteTerminal. Note that CiscoProvider must be in IN_SERVICE state, otherwise CiscoUnregistrationException about InvalidStateException will be thrown; or if terminal is not registered in Extend mode by this application or the unregistration fails for any reason, CiscoUnregistrationException will be thrown. And if terminal type is not CiscoRemoteTerminal and if terminal is not a Cisco Unified Client Services Framework device, then MethodNotSupportedException will be thrown.

Inteface	Method	Description
int	getType()	This method returns the device type of Terminal, which can be one of the following contants. It will return CiscoTerminal.DEVICETYPE_UNKNOWN if type of device cannot be found or device is invalid. CiscoTerminal.DEVICETYPE_UNKNOWN CiscoTerminal.DEVICETYPE_ANALOG_PHONE CiscoTerminal.DEVICETYPE_CISCO_6901 CiscoTerminal.DEVICETYPE_CISCO_6911 CiscoTerminal.DEVICETYPE_CISCO_6921 CiscoTerminal.DEVICETYPE_CISCO_6941 CiscoTerminal.DEVICETYPE_CISCO_6945 CiscoTerminal.DEVICETYPE_CISCO_6961 CiscoTerminal.DEVICETYPE_CISCO_7906 CiscoTerminal.DEVICETYPE_TELECASTER_BID CiscoTerminal.DEVICETYPE_CISCO_7911 CiscoTerminal.DEVICETYPE_14 _BUTTON_SIDECAR CiscoTerminal.DEVICETYPE_7915_12 _BUTTON_SIDECAR CiscoTerminal.DEVICETYPE_7915_24 _BUTTON_SIDECAR CiscoTerminal.DEVICETYPE_7916_12 _BUTTON_SIDECAR CiscoTerminal.DEVICETYPE_7916_24 _BUTTON_SIDECAR CiscoTerminal.DEVICETYPE_CKEM_36_BUTTON CiscoTerminal.DEVICETYPE_CP7921 CiscoTerminal.DEVICETYPE_CISCO_7925 CiscoTerminal.DEVICETYPE_CISCO_7926 CiscoTerminal.DEVICETYPE_7931

Inteface	Method	Description
		CiscoTerminal.DEVICETYPE_IP_ CONFERENCE_PHONE CiscoTerminal.DEVICETYPE_CISCO_7936 CiscoTerminal.DEVICETYPE_CISCO_7937 CiscoTerminal.DEVICETYPE _TELECASTER_BUSINESS CiscoTerminal.DEVICETYPE_CISCO_7941 CiscoTerminal.DEVICETYPE_CISCO_7941G_GE CiscoTerminal.DEVICETYPE_CISCO_7942 CiscoTerminal.DEVICETYPE_CISCO_7945 CiscoTerminal.DEVICETYPE_TELECASTER_MGR CiscoTerminal.DEVICETYPE_CISCO_7961 CiscoTerminal.DEVICETYPE_CISCO_7961G_GE CiscoTerminal.DEVICETYPE_CISCO_7962 CiscoTerminal.DEVICETYPE_CISCO_7965 CiscoTerminal.DEVICETYPE_CISCO_7970 CiscoTerminal.DEVICETYPE_CISCO_7971 CiscoTerminal.DEVICETYPE_CISCO_7975 CiscoTerminal.DEVICETYPE_CISCO_7989 CiscoTerminal.DEVICETYPE_CISCO_8941 CiscoTerminal.DEVICETYPE_CISCO_8945 CiscoTerminal.DEVICETYPE_CISCO_8961 CiscoTerminal.DEVICETYPE_9951 CiscoTerminal.DEVICETYPE_CISCO_9971 CiscoTerminal.DEVICETYPE_ATA_186 CiscoTerminal.DEVICETYPE_CISCO_ATA_187 CiscoTerminal.DEVICETYPE_CISCO_CIUS CiscoTerminal.DEVICETYPE_CISCO_CIUS_SP CiscoTerminal.DEVICETYPE_CISCO _SOFTPHONE_SE_M CiscoTerminal.DEVICETYPE_CISCO_UNIFIED _COMMUNICATOR CiscoTerminal.DEVICETYPE_CISCO_UNIFIED _MOBILE_COMMUNICATOR CiscoTerminal.DEVICETYPE_CISCO_UNIFIED _COMMUNICATIONS_FOR_RTX

Inteface	Method	Description
		CiscoTerminal.DEVICETYPE_CLIENT _SERVICES_FRAMEWORK CiscoTerminal.DEVICETYPE_VGC_PHONE CiscoTerminal.DEVICETYPE_CTI_PORT CiscoTerminal.DEVICETYPE_CTI_ROUTE_POINT CiscoTerminal.DEVICETYPE_DEVICE_PILOT CiscoTerminal.DEVICETYPE_ISDN_BRI_PHONE CiscoTerminal.DEVICETYPE_CTI_REMOTE_DEVICE
String	getTypeName()	This method returns the device type name of Terminal (i.e. The display name of the device type). It will return an empty string if device type name cannot be found, or return null if device is invalid.
CiscoTerminal	getCiscoMultiMediaCapabilityInfo()	Returns CiscoMultiMediaCapabilityInfo, to indicate the multimedia capabilities of the terminal.
int	getHuntLogStatus() throws InvalidStateException	This method is used get the value of huntlogstatus of the terminal, it returns either CiscoTerminal.DEVICE_HUNT_LOGGED_IN, CiscoTerminal.DEVICE_HUNT_LOGGED_OUT or CiscoTerminal. DEVICE_HUNT_NOT_APPLICABLE This method throws InvalidStateException if it is invoked on the terminal which is out of service
void	setHuntLogStatus(int huntLogStatus) throws InvalidStateException, MethodNotSupportedException, InvalidArgumentException	This method is used set the value of huntLogStatus of the device, it can take CiscoTerminal.DEVICE_HUNT_LOGGED_IN, CiscoTerminal.DEVICE_HUNT_LOGGED_OUT values as parameters. This method throws InvalidStateException if it is invoked on the terminal which is out of service. It throws MethodNotSupportedException when it is invoked on unsupported devices like CTI Route Points, CTI Remote Device and Spark Remote Device and InvalidArgumentException when the arguments are other than 1(loggen_in) and 2(logged_out)

Inherited Fields

From Interface javax.telephony.Terminal

addCallObserver, addObserver, getAddresses, getCallObservers, getCapabilities, getName, getObservers, getProvider, getTerminalCapabilities, getTerminalConnections, removeCallObserver, removeObserver

From Interface com.cisco.jtapi.extensions.CiscoObjectContainer

Data Type

public static final int

CiscoTerminal.DEVICETYPE_UNKNOWN = 0

CiscoTerminal.DEVICETYPE_ANALOG_PHONE = 30027

CiscoTerminal.DEVICETYPE_12S = 4

CiscoTerminal.DEVICETYPE_CISCO_6901 = 547

CiscoTerminal.DEVICETYPE_CISCO_6911 = 548

CiscoTerminal.DEVICETYPE_CISCO_6921 = 495

CiscoTerminal.DEVICETYPE_CISCO_6941 = 496

CiscoTerminal.DEVICETYPE_CISCO_6945 = 564

CiscoTerminal.DEVICETYPE_CISCO_6961 = 497

CiscoTerminal.DEVICETYPE_CISCO_7906 = 369

CiscoTerminal.DEVICETYPE_TELECASTER_BID = 6

CiscoTerminal.DEVICETYPE_CISCO_7911 = 307

CiscoTerminal.DEVICETYPE_14_BUTTON_SIDECAR = 124

CiscoTerminal.DEVICETYPE_7915_12_BUTTON_SIDECAR = 227

CiscoTerminal.DEVICETYPE_7915_24_BUTTON_SIDECAR = 228

CiscoTerminal.DEVICETYPE_7916_12_BUTTON_SIDECAR = 229

CiscoTerminal.DEVICETYPE_7916_24_BUTTON_SIDECAR = 230

CiscoTerminal.DEVICETYPE_CKEM_36_BUTTON = 232

CiscoTerminal.DEVICETYPE_CP7921 = 365

CiscoTerminal.DEVICETYPE_CISCO_7925 = 484

CiscoTerminal.DEVICETYPE_CISCO_7926 = 577

CiscoTerminal.DEVICETYPE_7931 = 348

CiscoTerminal.DEVICETYPE_IP_CONFERENCE_PHONE = 9

CiscoTerminal.DEVICETYPE_CISCO_7936 = 30019

CiscoTerminal.DEVICETYPE_CISCO_7937 = 431

CiscoTerminal.DEVICETYPE_TELECASTER_BUSINESS = 8

CiscoTerminal.DEVICETYPE_CISCO_7941 = 115

CiscoTerminal.DEVICETYPE_CISCO_7941G_GE = 309

CiscoTerminal.DEVICETYPE_CISCO_7942 = 434

CiscoTerminal.DEVICETYPE_CISCO_7945 = 435

CiscoTerminal.DEVICETYPE_TELECASTER_MGR = 7

CiscoTerminal.DEVICETYPE_CISCO_7961 = 30018

CiscoTerminal.DEVICETYPE_CISCO_7961G_GE = 308

CiscoTerminal.DEVICETYPE_CISCO_7962 = 404

CiscoTerminal.DEVICETYPE_CISCO_7965 = 436

CiscoTerminal.DEVICETYPE_CISCO_7970 = 30006

CiscoTerminal.DEVICETYPE_CISCO_7971 = 119

CiscoTerminal.DEVICETYPE_CISCO_7975 = 437

CiscoTerminal.DEVICETYPE_CISCO_7989 = 302

CiscoTerminal.DEVICETYPE_CISCO_8941 = 586

CiscoTerminal.DEVICETYPE_CISCO_8945 = 585

CiscoTerminal.DEVICETYPE_CISCO_8961 = 540

CiscoTerminal.DEVICETYPE_9951 = 537

CiscoTerminal.DEVICETYPE_CISCO_9971 = 493

CiscoTerminal.DEVICETYPE_ATA_186 = 12

CiscoTerminal.DEVICETYPE_CISCO_ATA_187 = 550

CiscoTerminal.DEVICETYPE_CISCO_CIUS = 593

CiscoTerminal.DEVICETYPE_CISCO_CIUS_SP = 632

CiscoTerminal.DEVICETYPE_CISCO_SOFTPHONE_SE_M = 30016

CiscoTerminal.DEVICETYPE_CISCO_UNIFIED_COMMUNICATOR = 358

CiscoTerminal.DEVICETYPE_CISCO_UNIFIED_MOBILE_COMMUNICATOR = 468

CiscoTerminal.DEVICETYPE_CISCO_UNIFIED_COMMUNICATIONS_FOR_RTX = 648

CiscoTerminal.DEVICETYPE_CLIENT_SERVICES_FRAMEWORK = 503

CiscoTerminal.DEVICETYPE_VGC_PHONE = 10

CiscoTerminal.DEVICETYPE_CTI_PORT = 72

CiscoTerminal.DEVICETYPE_CTI_ROUTE_POINT = 73

CiscoTerminal.DEVICETYPE_DEVICE_PILOT = 71

CiscoTerminal.DEVICETYPE_ISDN_BRI_PHONE = 30028

CiscoTerminal.DEVICETYPE_CTI_REMOTE_DEVICE = 635

Related Documentation

See Terminal, CiscoMediaTerminal, Constant Field Values, and CiscoTermEvFilter.

CiscoTerminalConnection

The CiscoTerminalConnection interface extends the CallControlTerminalConnection interface with additional capabilities. Applications can use the getReason method to obtain the reason for the creation of this Connection.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.
8.5(1)	Two new methods, addMediaStream(String streamDN, String callingPartyNumber) and removeMediaStream(), are added.
	A new API, playtone(int toneType, int playToneDirection) is added.
9.0(1)	A new method, startRecording(int playToneDirection, int recordingInvocationType), is added. Two new constants, RECORDING_INVOCATION_TYPE_SILENT, and RECORDING_INVOCATION_TYPE_USER, are added.
10.0(1)	A new method, hold (String contentID) is added.

Superinterfaces

javax.telephony.callcontrol.CallControlTerminalConnection, CiscoObjectContainer, javax.telephony.TerminalConnection

Declaration

public interface CiscoTerminalConnection extends javax.telephony.callcontrol.CallControlTerminalConnection, CiscoObjectContainer

Fields

Table 222: Fields in CiscoTerminalConnection

Interface	Field	Description
static final int	CISCO_SELECTEDNONE	The the call is not selected.
static final int	CISCO_SELECTEDLOCAL	The call is selected.
static final int	CISCO_SELECTEDREMOTE	A passive TerminalConnection receives this select status if the call is selected by its shared line.

Interface	Field	Description
static int	RECORDING_INVOCATION_TYPE_SILENT	This constant is used when the application invokes silent recording invocation type. The call recording status is not reflected on the Cisco IP device display. Silent recording is the default behavior in releases prior to Release 9.0. If an application uses the startRecording(int playToneDirection) method that was introduced prior to Release 9.0, it will default to the RECORDING_INVOCATION_TYPE_SILENT invocation type.
static int	RECORDING_INVOCATION_TYPE_USER	This constant is used when the application invokes user recording invocation type. The call recording status is reflected on the Cisco IP device display. Applications can query the device for this recording type.

Inherited Fields

From Interface javax.telephony.callcontrol.CallControlTerminalConnection

BRIDGED, DROPPED, HELD, IDLE, INUSE, RINGING, TALKING, UNKNOWN

From Interface javax.telephony.TerminalConnection

ACTIVE, PASSIVE

Parameters

invocationType

The invocationType parameter allows an application to specify a recording invocation type. The parameter is passed as one of the constants RECORDING_INVOCATION_TYPE_SILENT or RECORDING_INVOCATION_TYPE_USER.

String contentID

A String representing a specific video. Max 128 characters.

New Error Codes

CTIERR_ILLEGAL_CALLSTATE

Occurs if the request is made on a TerminalConnection associated with an invalid call. The only valid state to invoke this request is ‘Connected’.

JTAPI throws InvalidStateException with description as “Line is not in a legal state to invoke command.”

CTIERR _CALL_DROPPED

Occurs if the request is made on a TerminalConnection associated with an invalid call.

JTAPI throws InvalidStateException with description as “Call is dropped”.

CTIERR_BIB_NOT_CONFIGURED

Occurs if the Built-In-Bridge (BIB) is not configured on the agent device.

JTAPI throws ResourceUnavailableException with a description as “Built in bridge not configured”.

CTIERR_RESOURCE_NOT_AVAILABLE

Occurs if the Bulit-In-Bridge (BIB) cannot be allocated for the request.

JTAPI throws ResourceUnavailableException with a description as “Resource Not Available”.

CTIERR_MEDIA_CONNECTION_FAILED

Occurs if the Bulit-In-Bridge (BIB) call fails to connect to the media.

JTAPI throws InvalidStateException with a description as “The connection to the media has failed”.

CTIERR_START_STREAM_MEDIA_FAILED

Occurs if there is a general failure with the Agent Greeting feature, that is not covered by any of the other error codes.

JTAPI throws InvalidStateException with a description as “Start streaming media request failed”.

CTIERR_STOP_STREAM_MEDIA_FAILED

Occurs if there is a general failure with the Agent Greeting feature, that is not covered by any of the other error codes.

JTAPI throws InvalidStateException with a description as “Stop streaming media request failed”.

CTIERR_REQUEST_ALREADY_PENDING

Occurs if an application attempts to invoke an Agent Greeting API while another request is made.

JTAPI throws InvalidStateException with a description as “The request was rejected because there is a similar request already pending”.

CTIERR_NO_STREAMING_MEDIA_SESSION

Occurs if an application attempts to invoke a stop request while there is no existing media stream to stop.
JTAPI throws InvalidStateException with a description as “There is no streaming media session active”.

CTIERR_EXISTING_STREAMING_MEDIA_SESSION

Occurs if an application attempts to invoke an Agent Greeting API while another request is made and accepted.

JTAPI throws InvalidStateException with a description as “There is an existing streaming media session”.

CTIERR_RECORDING_INVOCATION_TYPE_NOT_MATCHING

Occurs if an application attempts to stop an active recording, but specifies a recording type other than the recording type that was used to invoke the recording.

Methods

Table 223: Methods in CiscoTerminalConnection

Interface	Method	Description
boolean	getPrivacyStatus()	Returns the privacy status of the call on the terminal. This interface returns True if Privacy is on and False otherwise. Always check the TerminalConnection privacy status before displaying any information about the call at an applcation Terminal implementation.
int	getSelectStatus()	Returns the select status of the call on the terminal. Always check the select status of the TerminalConnection before performing any call-process operation for the call. Can be one of: • CiscoTerminalConnection.CISCO_SELECTEDNONE • CiscoTerminalConnection.CISCO_SELECTEDLOCAL • CiscoTerminalConnection.CISCO_SELECTEDREMOTE

Interface	Method	Description

void	startRecording(intplayToneDirection)	Starts recording a call. The system delivers CiscoTermConnRecordingStartEv and CiscoTermConnRecordingTargetInfoEv to the call observer when this method is successful.
		Pre-conditions
		• ((this.getTerminal()).getProvider()).getState() = = Provider.IN_SERVICE
		• this.getCallControlState() = = CallControlTerminalConnection.TALKING
		• ((CiscoProviderCapabilities) (this.getTerminal().getProvider(). getProviderCapabilities()).canRecord() = = TRUE
		• this.getConnection().getAddress(). getRecordingConfig(this.getTerminal()) = = CiscoAddress. APPLICATION_CONTROLLED__RECORDING
		Parameters
		• playToneDirection—Specifies whether to play a tone. Valid values are:
		  • CiscoCall. PLAYTONE_NOLOCAL_OR_REMOTE
		  • CiscoCall. PLAYTONE_LOCALONLY
		  • CiscoCall. PLAYTONE_REMOTEONLY
		  • CiscoCall. PLAYTONE_BOTHLOCALANDREMOTE
		Throws
		• javax.telephony. InvalidStateException—Either the Provider was not "in service" or the TerminalConnection is not in the "TALKING" state.
		• javax.telephony. PrivilegeViolationException—The application does not have the proper authority to invoke this method.
		• javax.telephony. ResourceUnavailableException—An internal resource that this method requires is not available.
		• javax.telephony. InvalidArgumentException—The value for playToneDirection is not valid.

void	startRecording(int playToneDirection, int invocationType)	This method is similar to the startRecording(int playToneDirection) method.

Interface	Method	Description
void	stopRecording(int invocationType)	This method is similar to the stopRecording() method. If an application attempts to stop an active recording, but specifies a recording type other than the recording type that the recording was invoked with, the request fails and an exception with error code CTIERR_RECORDING_INVOCATION _TYPE_NOT_MATCHING is thrown.
CiscoRecorderInfo	getCiscoRecorderInfo()	Returns CiscoRecorderInfo, which exposes the terminal name and address of the recorder or null if the call is not being recorded. The call control terminal connection must be in the talking state.
CiscoMonitor InitiatorInfo	getCiscoMonitorInitiatorInfo()	Returns CiscoMonitorInitiatorInfo or null if the call is not being monitored. The application can use this method on the terminal connection of the monitor target to get information about the monitor initiator or determine that there is no monitor session.
CiscoMonitor TargetInfo	getCiscoMonitorTargetInfo()	Returns CiscoMonitorTargetInfo or null. The application can use this method on the terminal connection of the monitor initiator to get information about the monitor target. This method returns null when called on a terminal connection of the monitor target or if there is no monitor session.
void	addMediaStream(String streamDN, String callingPartyNumber)	Sends a request to begin sending media to the TerminalConnection's associated Built-in-bridge (BIB). **Parameters** • String streamDN—Dialed Number (DN) for the IVR, CTI Port, or the device streaming the media to the call. • String callingPartyNumber—A string object that applications use to provide information to the IVR. This is subject to all the constraints of a DN, and is used to store the agent's DN. This is presented to the IVR as the calling party in the new call event. The IVR must have some application running that understands this information, and can only be retrieved from the new call event on the IVR.
void	removeMediaStream()	Sends a request to cease the playing of media to the TerminalConnection's associated BIB.

Interface	Method	Description
void	playTone(int toneType, int playToneDirection)	Plays tones at local or remote ends of the call. **Parameters** • int tonetype—One of the tones defined in CiscoTone interface. • int playToneDirection—Can be CiscoCall.PLAYTONE_LOCALONLY or CiscoCall.PLAYTONE_REMOTEONLY. **Throws** • InvalidStateException • InvalidArgumentException • PlatformException
void	hold (String contentID)	This interface puts a call on hold, and specifies a contentID that can be used to play video on hold, or other related features. Note that CiscoProvider must be in IN_SERVICE state. Also, the call control state needs to be in TALKING state. If not, InvalidStateException will be thrown. If the hold request fails, ResourceUnavailableException will be thrown. All other errors encountered will result in PlatformException to be thrown. **Parameters** String contentID—A String representing a specific video. Max 128 characters.

Inherited Methods

From Interface javax.telephony.callcontrol.CallControlTerminalConnection

getCallControlState, hold, join, leave, unhold

From Interface javax.telephony.TerminalConnection

answer, getCapabilities, getConnection, getState, getTerminal, getTerminalConnectionCapabilities, getObject, setObject

From Interface com.cisco.jtapi.extensions.CiscoObjectContainer

Related Documentation

See Constant Field Values.

CiscoTerminalObserver

Applications implement this interface to receive CiscoTermEv events such as CiscoRTPInputStartedEv and CiscoRTPInputStoppedEv when observing Terminals via the Terminal.addObserver method.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.TerminalObserver

Declaration

public interface CiscoTerminalObserver extends javax.telephony.TerminalObserver

Fields

None

Methods

None

Inherited Methods

From Interface javax.telephony.TerminalObserver

terminalChangedEvent

Related Documentation

See CiscoTermInServiceEv, CiscoTermOutOfServiceEv, CiscoRTPInputStartedEv, CiscoRTPInputStoppedEv, CiscoRTPOutputStartedEv, and CiscoRTPOutputStoppedEv.

CiscoTerminalProtocol

The CiscoTerminalProtocol event is a container for the constants that define protocol types.

Interface History

Cisco Unified Communications Manager Release	Description
3.x	Added the extension.

Superinterfaces

public interface CiscoTerminalProtocol

Fields

Table 224: Fields in CiscoTerminalProtocol

Interface	Field	Description
static int	PROTOCOL_NONE	This constant value returned by the getProtocol() interface on CiscoTerminal indicates that the protocol type for the CiscoTerminal is not known or not available.
static int	PROTOCOL_SCCP	This constant value returned by the getProtocol() interface on CiscoTerminal indicates that the protocol type for the CiscoTerminal is SCCP.
static int	PROTOCOL_SIP	This constant value returned by the getProtocol() interface on CiscoTerminal indicates that the protocol type for the CiscoTerminal is SIP.
static int	PROTOCOL_CTI_REMOTE_DEVICE	This constant value returned by the getProtocol() interface on CiscoTerminal indicates that the protocol type for the CiscoTerminal is CTI Remote Device.

Related Documentation

See also CiscoTerminal, Constant Field Values for more information.

CiscoTermInServiceEv

The CiscoTermInServiceEv event gets sent to the application's TerminalObservers to indicate that the CiscoTerminal is ready for operation.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoTermInServiceEv extends CiscoTermEv

Fields

Table 225: Fields in CiscoTermInServiceEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 226: Methods in CiscoTermInServiceEv

Interface	Method	Description
int	getSupportedEncoding()	Reports whether a terminal is UNICODE-capable by returning the supported encoding. The value of supported encoding may be any of the following constants: • CiscoTerminal.UNKNOWN_ENCODING • CiscoTerminal.ASCII_ENCODING • CiscoTerminal.UCS2UNICODE_ENCODING • CiscoTerminal.NOT_APPLICABLE Returns: An integer value for the supported encoding of this terminal.
int	getLocale()	Returns the locale that this Terminal supports. Returns int values defined in the CiscoLocales interface.
boolean	getDNDStatus()	Returns the current DND status to the application.Returns boolean dndStatus.
int	getDNDOption()	Returns the current DND option to the application. The DND option can be any of the following constants: • CiscoTerminal.DND_OPTION_NONE • CiscoTerminal.DND_OPTION_RINGER_OFF • CiscoTerminal.DND_OPTION_CALL_REJECT Returns int dndOption.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values and CiscoLocales

CiscoTermOutOfServiceEv

The CiscoTermOutOfServiceEv event gets sent to the TerminalObservers of an application to indicate that the CiscoTerminal is out-of-service.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoOutOfServiceEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoTermOutOfServiceEv extends CiscoTermEv, CiscoOutOfServiceEv

Fields

Table 227: Fields in CiscoTermOutOfServiceEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING,

META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN, CAUSE_CALLMANAGER_FAILURE, CAUSE_CTIMANAGER_FAILURE, CAUSE_DEVICE_FAILURE, CAUSE_DEVICE_RESTRICTED, CAUSE_DEVICE_UNREGISTERED, CAUSE_LINE_RESTRICTED, CAUSE_NOCALLMANAGER_AVAILABLE, CAUSE_REHOME_TO_HIGHER_PRIORITY_CM, CAUSE_REHOMING_FAILURE

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface com.cisco.jtapi.extensions.CiscoOutOfServiceEv

Methods

None

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermRegistrationFailedEv

Applications receive this event when TerminalRegistration fails at the provider. The error that getErrorCode() returns explains the problem. On receiving this event, the application should try to reregister the terminal.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.
8.5(1)	A new error code, CTI_SECURITY_NOT_ALLOWED, is added.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoTermRegistrationFailedEv extends CiscoTermEv

Fields

Table 228: Fields in CiscoTermRegistrationFailedEv

Interface	Field	Description
static final int	ID	None
static final int	MEDIA_CAPABILITY_MISMATCH	Registration failed because the Terminal is already registered with a different media capability. Try reregistering with the same capability.
static final int	MEDIA_ALREADY_TERMINATED_NONE	Registration failed because the Terminal is already registered with media termination type none. Try reregistering with Media termination type none.
static final int	MEDIA_ALREADY_TERMINATED_STATIC	Registration failed because the Terminal is already registered with static media termination. Static registration does not allow a second registration. Wait until the terminal unregisters.
static final int	MEDIA_ALREADY_TERMINATED_DYNAMIC	Registration failed because the Terminal is already registered with dynamic media termination. Try reregistering with dynamic media termination.

Interface	Field	Description
static final int	OWNER_NOT_ALIVE	Registration encountered a race condition while attempting to register the Terminal. Try registering the Terminal.
static final int	DB_INITIALIZATION_ERROR	A database initialization error occurred while registering a Terminal. Try registering the Terminal.
static final int	UNKNOWN	Registration failed for an unknown internal reason. Try to reregister the Terminal.
static final int	IP_ADDRESSING_MODE_MISMATCH	Registration failed due to unsupported IP Addressing mode Try to register the Terminal with correct IP Addressing Mode.
public static	CTI_SECURITY_NOT_ALLOWED	The application registers a device in secured mode but eventually is rejected with the error code.

Inherited Fields

From Inteface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Inteface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 229: Methods in CiscoTermRegistrationFailedEv

Interface	Method	Description
int	getErrorCode()	Returns the error code for this exception, as an integer.

Inherited Methods

From Inteface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Inteface javax.telephony.events.TermEv

getTerminal

From Inteface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermRemovedEv

The CiscoTermRemovedEv event gets sent to the provider observer of the application when a CiscoTerminal gets removed from the provider domain.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Declaration

public interface CiscoTermRemovedEv extends CiscoProvEv

Fields

Table 230: Fields in CiscoTermRemovedEv

Interface	Field
static final int	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 231: Methods in CiscoTermRemovedEv

Interface	Method	Description
javax.telephony.Terminal	getTerminal()	Returns the Terminal that was removed from the provider domain.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermRestrictedEv

Applications see the CiscoTermRestrictedEv event when a user restricts a Terminal from Cisco Unified Communications Manager administration after the application starts running. Applications will not be able to see events for restricted Terminals, or for addresses on those terminals. If a Terminal gets restricted while it is in the in-service state, applications receive this event, and the Terminal and the corresponding addresses move to the out-of-service state.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoProvEv, javax.telephony.events.Ev, javax.telephony.events.ProvEv

Declaration

public interface CiscoTermRestrictedEv extends CiscoProvEv

Fields

Table 232: Fields in CiscoTermRestrictedEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN,

META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 233: Methods in CiscoTermRestrictedEv

Interface	Method	Description
javax.telephony.Terminal	getTerminal	Returns the Terminal that has become restricted.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.ProvEv

getProvider

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values.

CiscoTermSnapshotCompletedEv

If an application comes up after a call is established between two endpoints, for mid-call monitoring the application needs to query Terminal.createSnapshot(). After the call events for all of the Addresses on the Terminal get delivered, the application will get CiscoTermSnapshotCompletedEv. To maintain backward compatibility, these events get generated only when the application enables the snapShotRTPEnabled filter in CiscoTermEvFilter.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoTermSnapshotCompletedEv extends CiscoTermEv

Fields

Table 234: Fields in CiscoTermSnapshotCompletedEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

None

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See CiscoTermEvFilter and Constant Field Values.

CiscoTermSnapshotEv

If an application comes up after a call is established between two endpoints, for mid-call monitoring the application needs to query Terminal.createSnapshot(). The snapshot event, CiscoTermSnapshotEv, gets sent and indicates whether the current media between the endpoints is secure. Applications could also query CiscoMediaCallSecurityIndicator to get the security indicator for a call; however, this event does not contain any KeyMaterial. If there are no calls on any of the lines on the Terminal, applications will only get CiscoTermSnapshotCompletedEv. To maintain backward compatibility, these events get generated only when the application enables the snapShotRTPEnabled filter in CiscoTermEvFilter.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

CiscoEv, CiscoTermEv, javax.telephony.events.Ev, javax.telephony.events.TermEv

Declaration

public interface CiscoTermSnapshotEv extends CiscoTermEv

Fields

Table 235: Fields in CiscoTermSnapshotEv

Interface	Field
staticint	ID

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 236: Methods in CiscoTermSnapshotEv

Interface	Method	Description
CiscoMediaCallSecurity Indicator[]	getCiscoMediaCallSecurityIndicator()	Returns the media security status for each active call on this device.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.TermEv

getTerminal

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See CiscoTermEvFilter and Constant Field Values.

CiscoTone

The CiscoTone interface defines CTI Tone constant codes. CiscoToneChangedEv provides the getTone() method to return one of these constants. Only the ZIPZIP tone type is exposed to applications.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.0(1)	Added a new extension.
8.5(1)	Enhanced to include the supported tones that are used as parameters for playTone() method.

Superinterfaces

public interface CiscoTone

Fields

Table 237: Fields in CiscoTone

Interface	Field	Description
Static int	ZIPZIP	This interface defines the integer value for the ZIPZIP tone. The CiscoToneChangedEv.getTone() interface returns an integer value for tone.
public static final int	ZIP	-
public static final int	CALLWAITINGTONE	-

See also CiscoToneChangedEv, Constant Field Values.

CiscoToneChangedEv

The CiscoToneChangedEv event indicates that a tone has been generated for this call. The CallControlCallObserver interface reports this event. Currently, this tone gets generated only because of the Forced Authorization Code (FAC) or Client Matter Code (CMC) features. CiscoToneChangedEv.getCiscoCause() returns CiscoCallEv.CAUSE_FAC_CMC_FEATURE if the tone gets generated because of FAC_CMC.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.events.CallEv, CiscoCallEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoToneChangedEv extends CiscoCallEv

Fields

Table 238: Fields in CiscoToneChangedEv

Interface	Field	Description
static final int	ID	None
static final int	FAC_REQUIRED	Indicates that a FAC must be entered using Connection.addToAddress(String) to extend the call. This applies only for FAC_CMC_FEATURE_ID.
static final int	CMC_REQUIRED	Indicates that a CMC must be entered using Connection.addToAddress(String) to extend the call. This applies only for FAC_CMC_FEATURE_ID.
static final int	FAC_CMC_REQUIRED	Indicates that both a FAC and a CMC must be entered using Connection.addToAddress(String) to extend the call. The application can enter either one String code at a time or both at same time, separated by a # (pound sign) character. This applies only for FAC_CMC_FEATURE_ID.

CAUSE_ACCESSINFORMATIONDISCARDED

, CAUSE_BARGE, CAUSE_BCBPRESENTLYAVAIL, CAUSE_BCNAUTHORIZED,
CAUSE_BEARERCAPNIMPL, CAUSE_CALLBEINGDELIVERED, CAUSE_CALLIDINUSE,
CAUSE_CALLMANAGER_FAILURE, CAUSE_CALLREJECTED, CAUSE_CALLSPLIT,
CAUSE_CHANTYPENIMPL, CAUSE_CHANUNACCEPTABLE, CAUSE_CTICCMSIP400BADREQUEST,
CAUSE_CTICCMSIP401UNAUTHORIZED, CAUSE_CTICCMSIP402PAYMENTREQUIRED,
CAUSE_CTICCMSIP403FORBIDDEN, CAUSE_CTICCMSIP404NOTFOUND,
CAUSE_CTICCMSIP405METHODNOTALLOWED, CAUSE_CTICCMSIP406NOTACCEPTABLE,
CAUSE_CTICCMSIP407PROXYAUTHENTICATIONREQUIRED,
CAUSE_CTICCMSIP408REQUESTTIMEOUT, CAUSE_CTICCMSIP410GONE,
CAUSE_CTICCMSIP411LENGTHREQUIRED, CAUSE_CTICCMSIP413REQUESTENTITYTOOLONG,
CAUSE_CTICCMSIP414REQUESTURITOOLONG,
CAUSE_CTICCMSIP415UNSUPPORTEDMEDIATYPE,
CAUSE_CTICCMSIP416UNSUPPORTEDURISCHEME, CAUSE_CTICCMSIP420BADEXTENSION,
CAUSE_CTICCMSIP421EXTENSTIONREQUIRED, CAUSE_CTICCMSIP423INTERVALTOOBRIEF,
CAUSE_CTICCMSIP480TEMPORARILYUNAVAILABLE,
CAUSE_CTICCMSIP481CALLLEGDOESNOTEXIST, CAUSE_CTICCMSIP482LOOPDETECTED,
CAUSE_CTICCMSIP483TOOMANYHOOPS, CAUSE_CTICCMSIP484ADDRESSINCOMPLETE,
CAUSE_CTICCMSIP485AMBIGUOUS, CAUSE_CTICCMSIP486BUSYHERE,
CAUSE_CTICCMSIP487REQUESTTERMINATED, CAUSE_CTICCMSIP488NOTACCEPTABLEHERE,
CAUSE_CTICCMSIP491REQUESTPENDING, CAUSE_CTICCMSIP493UNDECIPHERABLE,
CAUSE_CTICCMSIP500SERVERINTERNALERROR, CAUSE_CTICCMSIP501NOTIMPLEMENTED,
CAUSE_CTICCMSIP502BADGATEWAY, CAUSE_CTICCMSIP503SERVICEUNAVAILABLE,
CAUSE_CTICCMSIP504SERVERTIMEOUT, CAUSE_CTICCMSIP505SIPVERSIONNOTSUPPORTED,
CAUSE_CTICCMSIP513MESSAGETOOLARGE, CAUSE_CTICCMSIP600BUSYEVERYWHERE,
CAUSE_CTICCMSIP603DECLINE, CAUSE_CTICCMSIP604DOESNOTEXISTANYWHERE,
CAUSE_CTICCMSIP606NOTACCEPTABLE, CAUSE_CTICONFERENCEFULL,
CAUSE_CTIDEVICENOTPREEMPTABLE, CAUSE_CTIDROPCONFEREE,
CAUSE_CTIMANAGER_FAILURE, CAUSE_CTIPRECEDENCECALLBLOCKED,
CAUSE_CTIPRECEDENCELEVELEXCEEDED, CAUSE_CTIPRECEDENCEOUTOFBANDWIDTH,
CAUSE_CTIPREEMPTFORREUSE, CAUSE_CTIPREEMPTNOREUSE,
CAUSE_DESTINATIONOUTOFORDER, CAUSE_DESTNUMMISSANDDCNOTSUB, CAUSE_DPARK,
CAUSE_DPARK_REMINDER, CAUSE_DPARK_UNPARK, CAUSE_EXCHANGEROUTINGERROR,
CAUSE_FAC_CMC, CAUSE_FACILITYREJECTED, CAUSE_IDENTIFIEDCHANDOESNOTEXIST,
CAUSE_IENIMPL, CAUSE_INBOUNDBLINDTRANSFER, CAUSE_INBOUNDCONFERENCE,
CAUSE_INBOUNDTRANSFER, CAUSE_INCOMINGCALLBARRED,
CAUSE_INCOMPATABLEDDESTINATION, CAUSE_INTERWORKINGUNSPECIFIED,
CAUSE_INVALIDCALLREFVALUE, CAUSE_INVALIDIECONTENTS,
CAUSE_INVALIDMESSAGEUNSPECIFIED, CAUSE_INVALIDNUMBERFORMAT,
CAUSE_INVALIDTRANSITNETSEL, CAUSE_MANDATORYIEMISSING,
CAUSE_MSGNCOMPATABLEWCS, CAUSE_MSGTYPENCOMPATWCS, CAUSE_MSGTYPENIMPL,
CAUSE_NETOUTOFORDER, CAUSE_NOANSWERFROMUSER, CAUSE_NOCALLSUSPENDED,
CAUSE_NOCIRCAVAIL, CAUSE_NOERROR, CAUSE_NONSELECTEDUSERCLEARING,
CAUSE_NORMALCALLCLEARING, CAUSE_NORMALUNSPECIFIED,
CAUSE_NOROUTETODDESTINATION, CAUSE_NOROUTETOTRANSITNET,
CAUSE_NOUSERRESPONDING, CAUSE_NUMBERCHANGED,
CAUSE_ONLYRDIVEARERCAPAVAIL, CAUSE_OUTBOUNDCONFERENCE,
CAUSE_OUTBOUNDTRANSFER, CAUSE_OUTOFBANDWIDTH,
CAUSE_PROTOCOLERRORUNSPECIFIED, CAUSE_QSIG_PR, CAUSE_QUALOFSERVNAVAIL,
CAUSE_QUIET_CLEAR, CAUSE_RECOVERYONTIMEREXPIRY, CAUSE_REDIRECTED,
CAUSE_REQCALLIDHASBEENCLEARED, CAUSE_REQCIRCNAVIL, CAUSE_REQFACILITYNIMPL,

CAUSE_REQFACILITYNOTSUBSCRIBED, CAUSE_RESOURCESNAVAIL, CAUSE_RESPONSETOSTATUSENQUIRY, CAUSE_SERVNOTAVAILUNSPECIFIED, CAUSE_SERVOPERATIONVIOLATED, CAUSE_SERVOROPTNAVAILORIMPL, CAUSE_SUBSCRIBERABSENT, CAUSE_SUSPCALLBUTNOTTHISONE, CAUSE_SWITCHINGEQUIPMENTCONGESTION, CAUSE_TEMPORARYFAILURE, CAUSE_UNALLOCATEDNUMBER, CAUSE_USERBUSY

Inherited Fields

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface com.cisco.jtapi.extensions.CiscoCallEv

Methods

Table 239: Methods in CiscoToneChangedEv

Interface	Method	Description
int	getTone()	Returns the generated tone type from CiscoTone.
int	getWhichCodeRequired()	Returns which codes are required for the dialed DN. The codeRequired may be one of the following: • CiscoToneChangedEv.FAC_REQUIRED • CiscoToneChangedEv.CMC_REQUIRED • CiscoToneChangedEv.FAC_CMC_REQUIRED

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface com.cisco.jtapi.extensions.CiscoCallEv

Related Documentation

See Constant Field Values.

CiscoTransferEndEv

The CiscoTransferEndEv event indicates that a transfer operation has completed. This event gets reported via the CallControlCallObserver interface.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.events.CallEv, CiscoCallEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoTransferEndEv extends CiscoCallEv

Fields

None

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoCallEv

CAUSE_ACCESSINFORMATIONDISCARDED, CAUSE_BARGE, CAUSE_BCBPRESENTLYAVAIL, CAUSE_BCNAUTHORIZED, CAUSE_BEARERCAPNIMPL, CAUSE_CALLBEINGDELIVERED, CAUSE_CALLIDINUSE, CAUSE_CALLMANAGER_FAILURE, CAUSE_CALLREJECTED, CAUSE_CALLSPLIT, CAUSE_CHANTYPENIMPL, CAUSE_CHANUNACCEPTABLE, CAUSE_CTICCMSIP400BADREQUEST, CAUSE_CTICCMSIP401UNAUTHORIZED, CAUSE_CTICCMSIP402PAYMENTREQUIRED, CAUSE_CTICCMSIP403FORBIDDEN, CAUSE_CTICCMSIP404NOTFOUND, CAUSE_CTICCMSIP405METHODNOTALLOWED, CAUSE_CTICCMSIP406NOTACCEPTABLE, CAUSE_CTICCMSIP407PROXYAUTHENTICATIONREQUIRED, CAUSE_CTICCMSIP408REQUESTTIMEOUT, CAUSE_CTICCMSIP410GONE, CAUSE_CTICCMSIP411LENGTHREQUIRED, CAUSE_CTICCMSIP413REQUESTENTITYTOOLONG, CAUSE_CTICCMSIP414REQUESTURITOOLONG, CAUSE_CTICCMSIP415UNSUPPORTEDMEDIATYPE, CAUSE_CTICCMSIP416UNSUPPORTEDURISCHEME, CAUSE_CTICCMSIP420BADEXTENSION, CAUSE_CTICCMSIP421EXTENSTIONREQUIRED, CAUSE_CTICCMSIP423INTERVALTOOBRIEF, CAUSE_CTICCMSIP480TEMPORARILYUNAVAILABLE, CAUSE_CTICCMSIP481CALLLEGDOESNOTEXIST, CAUSE_CTICCMSIP482LOOPDETECTED, CAUSE_CTICCMSIP483TOOMANYHOOPS, CAUSE_CTICCMSIP484ADDRESSINCOMPLETE, CAUSE_CTICCMSIP485AMBIGUOUS, CAUSE_CTICCMSIP486BUSYHERE, CAUSE_CTICCMSIP487REQUESTTERMINATED, CAUSE_CTICCMSIP488NOTACCEPTABLEHERE, CAUSE_CTICCMSIP491REQUESTPENDING, CAUSE_CTICCMSIP493UNDECIPHERABLE, CAUSE_CTICCMSIP500SERVERINTERNALERROR, CAUSE_CTICCMSIP501NOTIMPLEMENTED, CAUSE_CTICCMSIP502BADGATEWAY, CAUSE_CTICCMSIP503SERVICEUNAVAILABLE, CAUSE_CTICCMSIP504SERVERTIMEOUT, CAUSE_CTICCMSIP505SIPVERSIONNOTSUPPORTED, CAUSE_CTICCMSIP513MESSAGETOOLARGE, CAUSE_CTICCMSIP600BUSYEVERYWHERE, CAUSE_CTICCMSIP603DECLINE, CAUSE_CTICCMSIP604DOESNOTEXISTANYWHERE, CAUSE_CTICCMSIP606NOTACCEPTABLE, CAUSE_CTICONFERENCEFULL, CAUSE_CTIDEVICENOTPREEMPTABLE, CAUSE_CTIDROPCONFEREE, CAUSE_CTIMANAGER_FAILURE, CAUSE_CTIPRECEDENCECALLBLOCKED, CAUSE_CTIPRECEDENCELEVELEXCEEDED, CAUSE_CTIPRECEDENCEOUTOFBANDWIDTH, CAUSE_CTIPREEMPTFORREUSE, CAUSE_CTIPREEMPTNOREUSE, CAUSE_DESTINATIONOUTOFORDER, CAUSE_DESTNUMMISSANDDCNOTSUB, CAUSE_DPARK, CAUSE_DPARK_REMINDER, CAUSE_DPARK_UNPARK, CAUSE_EXCHANGEROUTINGERROR, CAUSE_FAC_CMC, CAUSE_FACILITYREJECTED, CAUSE_IDENTIFIEDCHANDOESNOTEXIST, CAUSE_IENIMPL, CAUSE_INBOUNDBLINDTRANSFER, CAUSE_INBOUNDCONFERENCE, CAUSE_INBOUNDTRANSFER, CAUSE_INCOMINGCALLBARRED, CAUSE_INCOMPATABLEDDESTINATION, CAUSE_INTERWORKINGUNSPECIFIED, CAUSE_INVALIDCALLREFVALUE, CAUSE_INVALIDIECONTENTS, CAUSE_INVALIDMESSAGEUNSPECIFIED, CAUSE_INVALIDNUMBERFORMAT, CAUSE_INVALIDTRANSITNETSEL, CAUSE_MANDATORYIEMISSING, CAUSE_MSGNCOMPATABLEWCS, CAUSE_MSGTYPENCOMPATWCS, CAUSE_MSGTYPENIMPL, CAUSE_NETOUTOFORDER, CAUSE_NOANSWERFROMUSER, CAUSE_NOCALLSUSPENDED, CAUSE_NOCIRCAVAIL, CAUSE_NOERROR, CAUSE_NONSELECTEDUSERCLEARING, CAUSE_NORMALCALLCLEARING, CAUSE_NORMALUNSPECIFIED, CAUSE_NOROUTETODDESTINATION, CAUSE_NOROUTETOTRANSITNET, CAUSE_NOUSERRESPONDING, CAUSE_NUMBERCHANGED, CAUSE_ONLYRDIVEARERCAPAVAIL, CAUSE_OUTBOUNDCONFERENCE,

CAUSE_OUTBOUNDTRANSFER, CAUSE_OUTOFBANDWIDTH, CAUSE_PROTOCOLERRORUNSPECIFIED, CAUSE_QSIG_PR, CAUSE_QUALOFSERVNAVAIL, CAUSE_QUIET_CLEAR, CAUSE_RECOVERYONTIMEREXPIRY, CAUSE_REDIRECTED, CAUSE_REQCALLIDHASBEENCLEARED, CAUSE_REQCIRCNAVIL, CAUSE_REQFACILITYNIMPL, CAUSE_REQFACILITYNOTSUBSCRIBED, CAUSE_RESOURCESNAVAIL, CAUSE_RESPONSETOSTATUSENQUIRY, CAUSE_SERVNOTAVAILUNSPECIFIED, CAUSE_SERVOPERATIONVIOLATED, CAUSE_SERVOROPTNAVAILORIMPL, CAUSE_SUBSCRIBERABSENT, CAUSE_SUSPCALLBUTNOTTHISONE, CAUSE_SWITCHINGEQUIPMENTCONGESTION, CAUSE_TEMPORARYFAILURE, CAUSE_UNALLOCATEDNUMBER, CAUSE_USERBUSY

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 240: Methods in CiscoTransferEndEv

Interface	Method	Description
javax.telephony.Call	getTransferredCall()	Returns the call that has been transferred. This call is in the Call.INVALID state.
javax.telephony.Call	getFinalCall()	Returns the call that remains active after the transfer completes.
javax.telephony.TerminalConnection	getTransferController()	Returns the ACTIVE TerminalConnection that currently acts as the transfer controller for the final call. When the transferController is a SharedLine, there will be multiple TerminalConnection objects. This method returns the ACTIVE TerminalConnection; however, if the application is not observing the ACTIVE TerminalConnection, this method will return one of the PASSIVE TerminalConnection objects.

Interface	Method	Description
javax.telephony.TerminalConnection[]	getTransferControllers()	Returns the list of TerminalConnection objects that currently act as the transfer controller for the final call. When the transferController is not a SharedLine, there will be only one TerminalConnection in the list. This method returns null if there is no observer on the transfer controller.
javax.telephony.Address	getTransferControllerAddress()	Returns the address that currently acts as the transfer controller for the final call.
boolean	isSuccess()	Returns true if the transfer is successful, or false otherwise.

Inherited Methods

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface com.cisco.jtapi.extensions.CiscoCallEv

getCiscoCause, getCiscoFeatureReason

Related Documentation

See Constant Field Values and getTransferControllers().

CiscoTransferStartEv

The CiscoTransferStartEv event indicates that a transfer operation has started. This event gets reported via the CallControlCallObserver interface.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.

Superinterfaces

javax.telephony.events.CallEv, CiscoCallEv, CiscoEv, javax.telephony.events.Ev

Declaration

public interface CiscoTransferStartEv extends CiscoCallEv

Fields

Table 241: Fields in CiscoTransferStartEv

Interface	Field
static final int	ID

Inherited Fields

From Interface com.cisco.jtapi.extensions.CiscoCallEv

CAUSE_ACCESSINFORMATIONDISCARDED, CAUSE_BARGE, CAUSE_BCBPRESENTLYAVAIL, CAUSE_BCNAUTHORIZED, CAUSE_BEARERCAPNIMPL, CAUSE_CALLBEINGDELIVERED, CAUSE_CALLIDINUSE, CAUSE_CALLMANAGER_FAILURE, CAUSE_CALLREJECTED, CAUSE_CALLSPLIT, CAUSE_CHANTYPENIMPL, CAUSE_CHANUNACCEPTABLE, CAUSE_CTICCMSIP400BADREQUEST, CAUSE_CTICCMSIP401UNAUTHORIZED, CAUSE_CTICCMSIP402PAYMENTREQUIRED, CAUSE_CTICCMSIP403FORBIDDEN, CAUSE_CTICCMSIP404NOTFOUND, CAUSE_CTICCMSIP405METHODNOTALLOWED, CAUSE_CTICCMSIP406NOTACCEPTABLE, CAUSE_CTICCMSIP407PROXYAUTHENTICATIONREQUIRED, CAUSE_CTICCMSIP408REQUESTTIMEOUT, CAUSE_CTICCMSIP410GONE, CAUSE_CTICCMSIP411LENGTHREQUIRED, CAUSE_CTICCMSIP413REQUESTENTITYTOOLONG, CAUSE_CTICCMSIP414REQUESTURITOOLONG, CAUSE_CTICCMSIP415UNSUPPORTEDMEDIATYPE, CAUSE_CTICCMSIP416UNSUPPORTEDURISCHEME, CAUSE_CTICCMSIP420BADEXTENSION, CAUSE_CTICCMSIP421EXTENSTIONREQUIRED, CAUSE_CTICCMSIP423INTERVALTOOBRIEF, CAUSE_CTICCMSIP480TEMPORARILYUNAVAILABLE, CAUSE_CTICCMSIP481CALLLEGDOESNOTEXIST, CAUSE_CTICCMSIP482LOOPDETECTED, CAUSE_CTICCMSIP483TOOMANYHOOPS, CAUSE_CTICCMSIP484ADDRESSINCOMPLETE, CAUSE_CTICCMSIP485AMBIGUOUS, CAUSE_CTICCMSIP486BUSYHERE, CAUSE_CTICCMSIP487REQUESTTERMINATED, CAUSE_CTICCMSIP488NOTACCEPTABLEHERE, CAUSE_CTICCMSIP491REQUESTPENDING, CAUSE_CTICCMSIP493UNDECIPHERABLE, CAUSE_CTICCMSIP500SERVERINTERNALERROR, CAUSE_CTICCMSIP501NOTIMPLEMENTED, CAUSE_CTICCMSIP502BADGATEWAY, CAUSE_CTICCMSIP503SERVICEUNAVAILABLE, CAUSE_CTICCMSIP504SERVERTIMEOUT, CAUSE_CTICCMSIP505SIPVERSIONNOTSUPPORTED, CAUSE_CTICCMSIP513MESSAGETOOLARGE, CAUSE_CTICCMSIP600BUSYEVERYWHERE, CAUSE_CTICCMSIP603DECLINE, CAUSE_CTICCMSIP604DOESNOTEXISTANYWHERE, CAUSE_CTICCMSIP606NOTACCEPTABLE, CAUSE_CTICONFERENCEFULL, CAUSE_CTIDEVICENOTPREEMPTABLE, CAUSE_CTIDROPCONFEREE, CAUSE_CTIMANAGER_FAILURE, CAUSE_CTIPRECEDENCECALLBLOCKED,

CAUSE_CTIPRECEDENCELEVELEXCEEDED, CAUSE_CTIPRECEDENCEOUTOFBANDWIDTH, CAUSE_CTIPREEMPTFORREUSE, CAUSE_CTIPREEMPTNOREUSE, CAUSE_DESTINATIONOUTOFORDER, CAUSE_DESTNUMMISSANDDCNOTSUB, CAUSE_DPARK, CAUSE_DPARK_REMINDER, CAUSE_DPARK_UNPARK, CAUSE_EXCHANGEROUTINGERROR, CAUSE_FAC_CMC, CAUSE_FACILITYREJECTED, CAUSE_IDENTIFIEDCHANDOESNOTEXIST, CAUSE_IENIMPL, CAUSE_INBOUNDBLINDTRANSFER, CAUSE_INBOUNDCONFERENCE, CAUSE_INBOUNDTRANSFER, CAUSE_INCOMINGCALLBARRED, CAUSE_INCOMPATABLEDDESTINATION, CAUSE_INTERWORKINGUNSPECIFIED, CAUSE_INVALIDCALLREFVALUE, CAUSE_INVALIDIECONTENTS, CAUSE_INVALIDMESSAGEUNSPECIFIED, CAUSE_INVALIDNUMBERFORMAT, CAUSE_INVALIDTRANSITNETSEL, CAUSE_MANDATORYIEMISSING, CAUSE_MSGNCOMPATABLEWCS, CAUSE_MSGTYPENCOMPATWCS, CAUSE_MSGTYPENIMPL, CAUSE_NETOUTOFORDER, CAUSE_NOANSWERFROMUSER, CAUSE_NOCALLSUSPENDED, CAUSE_NOCIRCAVAIL, CAUSE_NOERROR, CAUSE_NONSELECTEDUSERCLEARING, CAUSE_NORMALCALLCLEARING, CAUSE_NORMALUNSPECIFIED, CAUSE_NOROUTETODDESTINATION, CAUSE_NOROUTETOTRANSITNET, CAUSE_NOUSERRESPONDING, CAUSE_NUMBERCHANGED, CAUSE_ONLYRDIVEARERCAPAVAIL, CAUSE_OUTBOUNDCONFERENCE, CAUSE_OUTBOUNDTRANSFER, CAUSE_OUTOFBANDWIDTH, CAUSE_PROTOCOLERRORUNSPECIFIED, CAUSE_QSIG_PR, CAUSE_QUALOFSERVNAVAIL, CAUSE_QUIET_CLEAR, CAUSE_RECOVERYONTIMEREXPIRY, CAUSE_REDIRECTED, CAUSE_REQCALLIDHASBEENCLEARED, CAUSE_REQCIRCNAVIL, CAUSE_REQFACILITYNIMPL, CAUSE_REQFACILITYNOTSUBSCRIBED, CAUSE_RESOURCESNAVAIL, CAUSE_RESPONSETOSTATUSENQUIRY, CAUSE_SERVNOTAVAILUNSPECIFIED, CAUSE_SERVOPERATIONVIOLATED, CAUSE_SERVOROPTNAVAILORIMPL, CAUSE_SUBSCRIBERABSENT, CAUSE_SUSPCALLBUTNOTTHISONE, CAUSE_SWITCHINGEQUIPMENTCONGESTION, CAUSE_TEMPORARYFAILURE, CAUSE_UNALLOCATEDNUMBER, CAUSE_USERBUSY

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

From Interface javax.telephony.events.Ev

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_RESOURCES_NOT_AVAILABLE, CAUSE_SNAPSHOT, CAUSE_UNKNOWN, META_CALL_ADDITIONAL_PARTY, META_CALL_ENDING, META_CALL_MERGING, META_CALL_PROGRESS, META_CALL_REMOVING_PARTY, META_CALL_STARTING, META_CALL_TRANSFERRING, META_SNAPSHOT, META_UNKNOWN

Methods

Table 242: Methods in CiscoTransferStartEv

Interface	Method	Description
javax.telephony.Call	getTransferredCall()	Returns the call that will be transferred.
javax.telephony.Call	getFinalCall()	Returns the call that will remain active after the transfer completes.
javax.telephony.TerminalConnection	getTransferController()	Returns the ACTIVE TerminalConnection that currently acts as the transfer controller for the final call. When the transferController is a SharedLine, there will be multiple TerminalConnection objects. This method returns the ACTIVE TerminalConnection; however, if the application is not observing the ACTIVE TerminalConnection, this method will return one of the PASSIVE TerminalConnection objects.
javax.telephony.TerminalConnection[]	getTransferControllers()	Returns a list of TerminalConnection objects that currently act as the transfer controller for the final call. When the transferController is not a SharedLine, there will be only TerminalConnection in the list. This method returns null if there is no observer on the transfer controller.
javax.telephony.Address	getTransferControllerAddress()	Returns the address that currently acts as the transfer controller for the final call.
String	getControllerTerminalName()	Returns the terminal names of the controllers across which transfer is done.

Inherited Methods

From Interface com.cisco.jtapi.extensions.CiscoCallEv

getCiscoCause, getCiscoFeatureReason

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

From Interface javax.telephony.events.CallEv

getCall

From Interface javax.telephony.events.Ev

getCause, getID, getMetaCode, getObserved, isNewMetaEvent

Related Documentation

See Constant Field Values and getTransferControllers().

CiscoUrlInfo

The CiscoUrlInfo object specifies the properties of the Uniform Resources Locator (URL) associated with a SIP endpoint.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(1 and 2)	Created history table to track changes.
9.0(1)	Two new fields, TRANSPORT_TYPE_UNKNOWN and TRANSPORT_TYPE_TLS, are added.

Declaration

public interface CiscoUrlInfo

Fields

Table 243: Fields in CiscoUrlInfo

Interface	Field	Description
static final int	TRANSPORT_TYPE_UNKNOWN	The endpoint is using an unknown transport type.
static final int	TRANSPORT_TYPE_UDP	The endpoint is using UDP.
static final int	TRANSPORT_TYPE_TCP	The endpoint is using TCP.
static final int	TRANSPORT_TYPE_TLS	The endpoint is using TLS.
static final int	URL_TYPE_UNKNOWN	The URL is of unknown type.
static final int	URL_TYPE_TEL	The URL is of type telephony.
static final int	URL_TYPE_SIP	The URL is of type SIP.

Methods

Table 244: Methods in CiscoUrlInfro

Interface	Method	Description
java.lang.String	getUser()	Returns the user name associated with the SIP endpoint as a string
java.lang.String	getHost()	Returns the host name associated with the SIP endpoint.
int	getPort()	Returns the port associated with the SIP endpoint.
int	getTransportType()	Returns the Transport Layer Protocol type that the SIP endpoint uses. The type is either CiscoUrlInfo.TRANSPORT_TYPE_UDP or CiscoUrlInfo.TRANSPORT_TYPE_TCP.
int	getUrlType()	This method returns the endpoint URL type. CiscoUrlInfo.URL_TYPE_UNKNOWN, CiscoUrlInfo.URL_TYPE_TEL, and CiscoUrlInfo.URL_TYPE_SIP are the possible return values.

Related Documentation

See Constant Field Values.

ComponentUpdater

The overloaded method is introduced which creates an updater log in the directory that is specified.

Interface History

Cisco Unified Communications Manager Release Number	Description
7.1(2)	Added in 7.1(2)

Declaration

public interface ComponentUpdater

Methods

The overloaded method is introduced which creates updater log in the directory specified.

Table 245: Methods in ComponentUpdater

Interface	Method	Description
ComponentUpdater	String tracePath	Returns the Consult Call for which consult operation is cancelled, if the consult call doesn't exist it will return NULL.

Related Documentation

See Constant Field Values.

ProviderPickupNotificationRegistrationClosedEv

ProviderPickupNotificationRegistrationClosedEvent is a new interface being added with Call Pickup feature development. This event is fired whenever something happens on the CUCM that results in a previous registration to observe a pickup group being made invalid. For example, removal of pickup group from the CUCM, or change in Pickup Number etc. Applications should look for these events and handle them accodringly.

Interface History

Cisco Unified Communications Manager Release Number	Description
8.0(1)	New interface

Declaration

public interface ProviderPickupNotificationRegistrationClosedEvent extends CiscoProvEvMethods

Methods

Table 246: Methods in ProviderPickupNotificationRegistrationClosedEv

Interface	Method	Description
String	getPickupGroupNumber()	This method returns the Pickup Group Number for the Pickup Group that is invalidated by the event.
String	getPickupGroupPartition()	This method returns the Pickup Group Partition for the Pickup Group that is invalidated by the event.
int	getReason()	This method returns the reason code explaining why this Pickup Group was invalidated.

New Reason Code

CTIERR_PICKUPGROUP_CHANGED

CTIERR_PICKUPGROUP_DELETED

Related Documentation

None

CiscoTermHuntLogStatusChangedEv

This is a new interface that has been introduced in Cisco JTAPI. The intention of this new interface is to notify the applications with event CiscoTermHuntLogStatusChangedEv whenever the value of hunt log status is changed, provided the filter is set to true on that particular terminal.

Declaration

Methods

Inteface	Method	Description
int	getHuntLogStatus()	This method is used get the value of huntlogstatus of the terminal, it returns either CiscoTerminal.DEVICE_HUNT_LOGGED_IN, CiscoTerminal.DEVICE_HUNT_LOGGED_OUT, or CiscoTerminal. DEVICE_HUNT_NOT_APPLICABLE

CiscoProvConnToLeastPriorCtiServerEv

Interface History

Cisco Unified Communications Manager Release Number	Description
14SU3	New interface.

Declaration

public interface CiscoProvConnToLeastPriorCtiServerEv extends CiscoProvEv.

Fields

Table 247: Fields in CiscoProvConnToLeastPriorCtiServerEv

Interface	Field	Description
Static int	ID	

Methods

Related Documentation

None

CiscoProvFallbackToPrimNwCompltdEv

Interface History

Cisco Unified Communications Manager Release Number	Description
14SU3	New interface.

Declaration

public interface CiscoProvFallbackToPrimNwCompltdEv extends CiscoProvEv.

Fields

Table 248: Fields in CiscoProvFallbackToPrimNwCompltdEv

Interface	Field	Description
Static int	ID	

Methods

Related Documentation

None

CiscoProvPrimNwReachableEv

Interface History

Cisco Unified Communications Manager Release Number	Description
14SU3	New interface.

Declaration

public interface CiscoProvPrimNwReachableEv extends CiscoProvEv.

Fields

None

Methods

Table 249: Methods in CiscoProvPrimNwReachableEv

Interface	Method	Description
String[]	getReachableCtiServers()	Returns a list of CTI Servers that are reachable after application fails over to the least priority CTI Server.

Related Documentation

None