CSTA

CSTADLL
Version 2.3.4
CSTADLL
Reference Manual

Objective Systems, Inc. version 2.3.4 February 2021

Author's Contact Information

Comments, suggestions, and inquiries regarding CSTADLL may be submitted via electronic mail to info@obj-sys.com.

Contents

Chapter 1

CSTAXMLEd4

CSTAXMLEd4 is a Microsoft .NET 4.5 DLL that allows client code to communicate with a PBX or UA device.

The DLL uses the following namespaces:

- `Com.Objsys.Csta.Devices`
- `Com.Objsys.Csta.Xmled4`

The `Com.Objsys.Csta.Devices` namespace contains classes that allow a caller to use specific PBX or UA devices.

The `Com.Objsys.Csta.Xmled(n)` namespaces contain classes that are specific to the indicated edition of XML CSTA. Most of these classes are generated by XBinder from the CSTA and session management (ECMA-354) XML schema specifications. These generated classes are not documented here, but you can consult the XBinder Java/C# User Guide for information about how XML schema constructions are translated into C# classes.

Each namespace also contains several classes that are not generated by ASN1C. These classes are the ones documented in this manual.

A typical way to use the DLL is to use the `PBXSession` class to set up the communication to the PBX or UA device via the constructor. If the PBX or UA will be sending asynchronous data, such as monitor packets, to the client, the `ClientCallback` or `XMLClientCallback` property can be used to define a callback method to receive the asynchronous data. If no callback method is defined, asynchronous data will be ignored.

If the PBX or UA will be sending Call Detail Records Report or Call Detail Records Notification messages to the client, the `CDRCallback` or `XMLCDRCallback` property can be used to define a callback method to receive the messages. If no callback method is defined, Call Detail messages will be ignored.

The CSTADLL kit includes some samples to guide you in writing your own code. The samples are evenly split between those implemented in C# and those implemented in Visual BASIC. Each language has samples for communicating with PBX devices that use BER CSTA and with PBX devices that use XML CSTA.

The classes and methods exposed by the DLL are probably sufficient to handle operations for most PBX or UA devices. But if needed, you can write a class of your own to handle operations for a PBX device that the software doesn't explicitly support. The sample NewPBX shows how this might be accomplished. This sample contains code for a small separate DLL that could be used to support a fictitious PBX device. The assumption in the sample is that this device uses standard messages for all operations except for the initial association messages. These messages are the ones that are most commonly different from one PBX to the next. The NewPBX sample shows how the EncodeACSEConnectionRequest() method within the GenericCSTAp2 class (for BER PBX devices) or the EncodeStartSession() method within the GenericXMLEd4 class (for XML PBX devices) can be overridden in a class that you can write. The override implementation handles the details that are specific to the device.

The DLL can log message traffic between a client program and the PBX or UA device if so desired. The logging is controlled by the `LoggingEnabled` property with the `PBXSessionHelper` class. The logging is off by default. Both of the provided sample clients enable the logging. The log file used is named `cstadll_<program>.log`, where <program> is the name of the executable image that is using the DLL. The location of the log file is the folder where the executable image resides. The default behavior is that if the log file grows to more then 5 Mb, it is copied to `cstadll_<program>.backup.log`, and a new log file is opened. If there is already a file with the backup file name, it is overwritten. That default size of 5 Mb can be modified by using the `MaxLogFileSize` property of the `PBXSessionHelper` class.

If your CSTADLL kit is licensed (i.e., not unlimited), then you will need to deploy your application with the DLLs Reprise.dll and rlm1212.dll that are in the kit. The file rlm1212.dll is a 32-bit native DLL as opposed to a .NET DLL. As such, if you build your code with a Makefile, you will need to use the `/platform:x86` qualifier to the `csc` or `vbc` command. If you build your code with a Visual Studio project, you will need to use x86 as the target platform instead of AnyCPU. These steps are to ensure proper interfacing to the native 32-bit rlm1212.dll. There is also a 64-bit version of rlm1212.dll available if you prefer to target the x64 platform.

Chapter 2

Namespace Documentation

2.1 Package Com.Objsys.Csta.Devices

Classes

- class UnifyOpenscapeVoice

2.1.1 Detailed Description

The namespace `Com.Objsys.Csta.Devices` contains classes that allow a caller to use specific PBX devices. The caller does not need to know what CSTA phase a device uses unless the device can accept messages formatted according to rules from more than one CSTA phase. In that case the class name ends with 'p(n)', where (n) is the number of the phase.

2.2 Package Com.Objsys.Csta.Xmled4

Classes

- class Constants
- class CSTAContext
- class CSTAResponseInfo
- class GenericXMLed4
- class LicenseException
- class LicenseOptions
- class PBXSession
- class PBXSessionException
- class PBXSessionHelper
- class PBXSessionHelperEd4
- class ResetSessionInfo
- class SocketState
- class uaSIPInvite
- class uaXMLed4

2.2.1 Detailed Description

The namespace `Com.Objsys.Csta.Xmled4` contains classes that are specific to XML CSTA edition 4. Most of these classes are generated by XBinder from the CSTA and session management (ECMA-354) XML schema specifications. These generated classes are not documented here, but you can consult the XBinder Java/C# User Guide for information about how XML schema constructions are translated into C# classes.

The namespace also contains several classes that are not generated by XBinder. These classes are the ones documented in this manual.

Chapter 3

Class Documentation

3.1 Constants Class Reference

Public Types

- enum ACSEMessageTypes
- enum CallbackInvocationMechanisms
- enum CommunicationTypes
- enum Encoding
- enum PBXModels
- enum XMLSessionMessageTypes

Public Attributes

- const long MAX_LOGFILE_SIZE = $(5 * 1024 * 1024)$

3.1.1 Detailed Description

The Constants class contains some helpful constant and enum definitions.

3.1.2 Member Enumeration Documentation

3.1.2.1 enum ACSEMessageTypes

Provides symbolic names for the ACSE message types.

3.1.2.2 enum CallbackInvocationMechanisms

Indicates how an asynchronous callback method should be invoked. This setting influences how the asynchronous callback methods for monitor event report messages, route messages, and Call Detail Record messages are invoked.

The value InvokeCallbackThenPostNextRead causes the callback method to be invoked before the next read from the PBX or UA is posted to the socket. This setting is the default. With this mechanism callback methods can be easily debugged because new packets from the PBX or UA won't be arriving while debugging of the method is in progress. This mechanism also ensures that messages from the PBX or UA will arrive in a predictable order.

The value PostNextReadThenInvokeCallback causes the callback method to be invoked after the next read from the PBX or UA is posted to the socket. Use of this mechanism is necessary if additional synchronous CSTA messages are going to be sent as part of a callback method's processing. If this mechanism is not used in such a case, the response to the CSTA message sent from the callback method will never be seen because no read to the socket was posted. With that said, however, use this mechanism with EXTREME caution. Because the read to the socket is posted before the event is handled, event n+1 may come in and get handled before event n. You may need to add code to ensure that events get handled in an expected order, if such code is even possible for your situation.

3.1.2.3 enum CommunicationTypes

Provides symbolic names for different ways of communicating with a PBX or UA. The values of this enum influence how each message exchange with a PBX or UA is handled.

3.1.2.4 enum Encoding

Provides symbolic names for the mechanisms for encoding CSTA messages.

3.1.2.5 enum PBXModels

Provides symbolic names for different PBX models.

3.1.2.6 enum XMLSessionMessageTypes

Provides symbolic names for the XML session management message types.

3.1.3 Member Data Documentation

3.1.3.1 const long MAX_LOGFILE_SIZE = (5 ∗ 1024 ∗ 1024)

Defines the maximum size, in bytes, that a log file is allowed to grow to before a new log file is opened.

3.2 CSTAContext Class Reference

Properties

- byte[ ] ResponseFromPBX `[get, set]`
- List< byte[ ]> ResponsesFromPBX `[get, set]`
- string XMLResponseFromPBX `[get, set]`
- List< string > XMLResponsesFromPBX `[get, set]`

3.2.1 Detailed Description

The CSTAContext class contains information needed to manage the interaction between the thread and the PBX.

3.2.2 Property Documentation

3.2.2.1 byte [ ] ResponseFromPBX `[get, set]`

See documentation for CSTAResponseInfo.ResponseFromPBX.

3.2.2.2 List<byte[ ]> ResponsesFromPBX `[get, set]`

See documentation for CSTAResponseInfo.ResponsesFromPBX.

3.2.2.3 string XMLResponseFromPBX `[get, set]`

See documentation for CSTAResponseInfo.XMLResponseFromPBX.

3.2.2.4 List<string> XMLResponsesFromPBX `[get, set]`

See documentation for CSTAResponseInfo.XMLResponsesFromPBX.

3.3 CSTAResponseInfo Class Reference

Properties

- byte[ ] ResponseFromPBX `[get, set]`
- List< byte[ ]> ResponsesFromPBX `[get, set]`
- List< byte[ ]> ResponsesFromUA `[get, set]`
- int StatusCode `[get, set]`
- string StatusMessage `[get, set]`
- string XMLResponseFromPBX `[get, set]`
- string XMLResponseFromUA `[get, set]`
- List< string > XMLResponsesFromPBX `[get, set]`

3.3.1 Detailed Description

Contains information about a PBX operation that was attempted.

3.3.2 Property Documentation

3.3.2.1 byte [ ] ResponseFromPBX `[get, set]`

Contains the response from the PBX for messages that generate a single atomic response, or the immediate acknowledgement response for messages that generate multiple data responses (e.g., Get Switching Function Devices). If a message that normally generates multiple response segments encounters an error (e.g., the PBX rejects the message), then the single error message returned by the PBX will be in this property; the ResponsesFromPBX property will be null.

For CSTA operations this property is simply a reference to the ResponseFromPBX property of the thread's CSTAContext object. If the value of that property changes, then the value of this property changes.

3.3.2.2 List<byte[ ]> ResponsesFromPBX `[get, set]`

Contains the responses from the PBX for messages that generate multiple response segments (e.g., Get Switching Function Devices). If such a message encounters an error (e.g., the PBX rejects the message), then the single error message returned by the PBX will be in the ResponseFromPBX property; this property will be null. In all cases the first response, which is the acknowledgement message from the PBX, will be in the ResponseFromPBX property.

This property is simply a reference to the ResponsesFromPBX property of the thread's CSTAContext object. If the value of that property changes, then the value of this property changes.

3.3.2.3 List<byte[ ]> ResponsesFromUA `[get, set]`

Contains the responses from the UA for messages that generate multiple response segments (e.g., Get Switching Function Devices). If such a message encounters an error (e.g., the UA rejects the message), then the single error message returned by the UA will be in the ResponseFromUA property; this property will be null. In all cases the first response, which is the acknowledgement message from the UA, will be in the ResponseFromUA property.

This property is simply a reference to the ResponsesFromPBX property of the thread's CSTAContext object. If the value of that property changes, then the value of this property changes.

3.3.2.4 int StatusCode `[get, set]`

A numeric status code. A value less than zero indicates that something went wrong during the attempted operation.

3.3.2.5 string StatusMessage `[get, set]`

Text containing information about a PBX operation that has completed, either successfully or not.

3.3.2.6 string XMLResponseFromPBX `[get, set]`

Contains the response from the PBX for messages that generate a single atomic XML response, or the immediate XML acknowledgement response for messages that generate multiple XML data responses (e.g., GetSwitchingFunctionDevices). If a message that normally generates multiple response segments encounters an error (e.g., the PBX rejects the message), then the single error message returned by the PBX will be in this property; the ResponsesFromPBX property will be null.

For CSTA operations this property is simply a reference to the XMLResponseFromPBX property of the thread's CSTAContext object. If the value of that property changes, then the value of this property changes.

3.3.2.7 string XMLResponseFromUA `[get, set]`

Contains the response from the UA for messages that generate a single atomic XML response, or the immediate XML acknowledgement response for messages that generate multiple XML data responses (e.g., GetSwitchingFunctionDevices). If a message that normally generates multiple response segments encounters an error (e.g., the UA rejects the message), then the single error message returned by the UA will be in this property; the ResponsesFromUA property will be null.

For CSTA operations this property is simply a reference to the XMLResponseFromPBX property of the thread's CSTAContext object. If the value of that property changes, then the value of this property changes.

3.3.2.8 List<string> XMLResponsesFromPBX `[get, set]`

Contains the responses from the PBX for messages that generate multiple XML response segments (e.g., GetSwitchingFunctionDevices). If such a message encounters an error (e.g., the PBX rejects the message), then the single error message returned by the PBX will be in the ResponseFromPBX property; this property will be null. In all cases the first response, which is the acknowledgement message from the PBX, will be in the ResponseFromPBX property.

This property is simply a reference to the XMLResponsesFromPBX property of the thread's CSTAContext object. If the value of that property changes, then the value of this property changes.

3.4 GenericXMLed4 Class Reference

Inherited by UnifyOpenscapeVoice, and uaXMLed4.

Public Member Functions

- virtual CSTAResponseInfo AnswerCall (ConnectionID callToAnswer, string deviceToLift)
- virtual CSTAResponseInfo AnswerCall (ConnectionID callToAnswer)
- virtual CSTAResponseInfo ClearConnection (ConnectionID connectionToClear)
- virtual CSTAResponseInfo ClearDoNotDisturb (string targetDevice)
- virtual CSTAResponseInfo ClearMessageWaiting (string targetDevice)
- virtual CSTAResponseInfo ConsultationCall (ConnectionID existingCall, string targetDevice)
- virtual string EncodeInvokeID (string initialMessage)
- GenericXMLed4 (PBXSession sessionObject)
- GenericXMLed4 (string pbxSystem, int port)
- virtual CSTAResponseInfo GetAgentState (string agentDevice)
- virtual CSTAResponseInfo GetDoNotDisturb (string targetDevice)
- virtual CSTAResponseInfo GetLogicalDevInfo (string targetDevice)
- virtual CSTAResponseInfo GetPhysicalDevInfo (string targetDevice)
- virtual CSTAResponseInfo GetSFDevices ()
- virtual CSTAResponseInfo GetSFDevices (ReqDeviceCategory deviceCategory)
- virtual CSTAResponseInfo HoldCall (ConnectionID callToHold)
- virtual CSTAResponseInfo MakeCall (string callingDevice, string calledDevice)
- virtual CSTAResponseInfo MonitorStart (ConnectionID callToMonitor)
- virtual CSTAResponseInfo MonitorStart (string deviceToMonitor)
- virtual CSTAResponseInfo MonitorStop (string crossRefID)
- virtual CSTAResponseInfo MonitorStopAtDevice (string monitoredDevice)
- virtual CSTAResponseInfo RequestSystemStatus ()
- virtual CSTAResponseInfo RetrieveCall (ConnectionID callToRetrieve)
- virtual CSTAResponseInfo RingDevice (string targetDevice, string targetRinger, long ringPattern)
- virtual CSTAResponseInfo SendData (IOCrossRefID xref, string text)
- virtual CSTAResponseInfo SendStoredCDR (string cdrCrossRefID)
- virtual CSTAResponseInfo SetAgentState (string agentDevice, ReqAgentState agentState)
- virtual CSTAResponseInfo SetAgentState (string agentDevice, ReqAgentState agentState, string agentID)
- virtual CSTAResponseInfo SetDisplay (string targetDevice, string text)
- virtual CSTAResponseInfo SetDoNotDisturb (string targetDevice)
- virtual CSTAResponseInfo SetMessageWaiting (string targetDevice)
- virtual CSTAResponseInfo SingleStepTransfer (ConnectionID callToTransfer, string transferToDevice)
- virtual CSTAResponseInfo SnapshotCall (ConnectionID callToSnapshot)
- virtual CSTAResponseInfo SnapshotDevice (string deviceToSnapshot)
- virtual CSTAResponseInfo StartCDRTransmission (CDRTransferMode transferMode)
- virtual CSTAResponseInfo StartDataPath (string targetDevice)
- virtual CSTAResponseInfo StartSession (string applicationID)
- virtual CSTAResponseInfo StartSession ()
- virtual CSTAResponseInfo StopCDRTransmission (string cdrCrossRefID)
- virtual CSTAResponseInfo StopDataPath (IOCrossRefID xref)
- virtual CSTAResponseInfo StopRing (string targetDevice, string targetRinger, long ringPattern)
- virtual CSTAResponseInfo StopSession ()
- virtual CSTAResponseInfo TransferCall (ConnectionID initiatedCall, ConnectionID originalCall)

Protected Member Functions

- virtual string EncodeAnswerCall (CSTAResponseInfo response, ConnectionID callToAnswer, string deviceToLift)
- virtual string EncodeAnswerCall (CSTAResponseInfo response, ConnectionID callToAnswer)
- virtual string EncodeClearConnection (CSTAResponseInfo response, ConnectionID connectionToClear)
- virtual string EncodeConsultationCall (CSTAResponseInfo response, ConnectionID existingCall, string targetDevice)
- virtual string EncodeGetAgentState (CSTAResponseInfo response, string targetDevice)
- virtual string EncodeGetDND (CSTAResponseInfo response, string targetDevice)
- virtual string EncodeGetLogicalDevInfo (CSTAResponseInfo response, string targetDevice)
- virtual string EncodeGetPhysicalDevInfo (CSTAResponseInfo response, string targetDevice)
- virtual string EncodeGetSFDevices (CSTAResponseInfo response, ReqDeviceCategory category)
- virtual string EncodeHoldCall (CSTAResponseInfo response, ConnectionID callToHold)
- virtual string EncodeMakeCall (CSTAResponseInfo response, string callingDevice, string calledDevice)
- virtual string EncodeMonitorStart (CSTAResponseInfo response, ConnectionID targetCall)
- virtual string EncodeMonitorStart (CSTAResponseInfo response, string targetDevice)
- virtual string EncodeMonitorStop (CSTAResponseInfo response, string xref)
- virtual string EncodeRequestSystemStatus (CSTAResponseInfo response)
- virtual string EncodeRetrieveCall (CSTAResponseInfo response, ConnectionID callToRetrieve)
- virtual string EncodeSendData (CSTAResponseInfo response, IOCrossRefID xref, string strText)
- virtual string EncodeSendStoredCDR (CSTAResponseInfo response, string cdrCrossRefID)
- virtual string EncodeSetAgentState (CSTAResponseInfo response, string agentDevice, ReqAgentState agentState)
- virtual string EncodeSetAgentState (CSTAResponseInfo response, string agentDevice, ReqAgentState agentState, string agentID)
- virtual string EncodeSetDisplay (CSTAResponseInfo response, string targetDevice, string text)
- virtual string EncodeSetDND (CSTAResponseInfo response, string targetDevice, bool dndOn)
- virtual string EncodeSetMsgWaiting (CSTAResponseInfo response, string targetDevice, bool indicatorOn)
- virtual string EncodeSetRingerStatus (CSTAResponseInfo response, string targetDevice, string targetRinger, RingMode rm, long ringPattern)
- virtual string EncodeSingleStepTransfer (CSTAResponseInfo response, ConnectionID callToTransfer, string transferToDevice)
- virtual string EncodeSnapshotCall (CSTAResponseInfo response, ConnectionID callToSnapshot)
- virtual string EncodeSnapshotDevice (CSTAResponseInfo response, string targetDevice)
- virtual string EncodeStartCDRTrans (CSTAResponseInfo response, CDRTransferMode transferMode)
- virtual string EncodeStartDataPath (CSTAResponseInfo response, string targetDevice)
- virtual string EncodeStartSession (CSTAResponseInfo response, string applicationID)
- virtual string EncodeStopCDRTrans (CSTAResponseInfo response, string cdrCrossRefID)
- virtual string EncodeStopDataPath (CSTAResponseInfo response, IOCrossRefID xref)
- virtual string EncodeStopSession (CSTAResponseInfo response)
- virtual string EncodeTransferCall (CSTAResponseInfo response, ConnectionID initiatedCall, ConnectionID originalCall)

Properties

- PBXSession SessionObject `[get]`
- CSTAContext ThreadContext `[get]`

3.4.1 Detailed Description

Implements CSTA phase 3 operations using XML edition 4. Note that most PBXes don't support all CSTA messages, so some methods in this class may result in an error status being returned by your PBX.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 GenericXMLed4 (string *pbxSystem*, int *port*)

Constructs an instance associated with the given PBX identifier and port.

Parameters

pbxSystem Well-known name or IP address of the PBX.

port Port on which the PBX listens for CSTA messages.

3.4.2.2 GenericXMLed4 (PBXSession *sessionObject*)

Constructs an instance associated with the given PBXSession object.

Parameters

sessionObject A PBXSession object.

3.4.3 Member Function Documentation

3.4.3.1 virtual CSTAResponseInfo AnswerCall (ConnectionID *callToAnswer*, string *deviceToLift*) `[virtual]`

Answers a call.

Parameters

callToAnswer ConnectionID of an existing call (such as initiated through MakeCall()).

deviceToLift The device (e.g., "800") that is to answer the call.

Returns

A CSTAResponseInfo object.

3.4.3.2 virtual CSTAResponseInfo AnswerCall (ConnectionID *callToAnswer*) `[virtual]`

Answers a call.

Parameters

callToAnswer The ConnectionID of the call to answer.

Returns

A CSTAResponseInfo object.

3.4.3.3 virtual CSTAResponseInfo ClearConnection (ConnectionID *connectionToClear*) `[virtual]`

Clears a connection.

Parameters

connectionToClear The ConnectionID of the connection to clear.

Returns

A CSTAResponseInfo object.

3.4.3.4 virtual CSTAResponseInfo ClearDoNotDisturb (string *targetDevice*) `[virtual]`

Turns off the Do Not Disturb functionality for a phone.

Parameters

targetDevice The device for which the Do Not Disturb functionality is to be turned off.

Returns

A CSTAResponseInfo object.

3.4.3.5 virtual CSTAResponseInfo ClearMessageWaiting (string *targetDevice*) `[virtual]`

Turns off the message waiting indicator on a device's display.

Parameters

targetDevice The device for which the indicator is to be turned off.

Returns

A CSTAResponseInfo object.

3.4.3.6 virtual CSTAResponseInfo ConsultationCall (ConnectionID *existingCall*, string *targetDevice*) `[virtual]`

Instruct the PBX to do a consultation call.

Parameters

existingCall The connection id of the call for which the consultation call will be made.

targetDevice Identifier (e.g., phone number) of the device that is the target of the consultation call.

Returns

A CSTAResponseInfo object.

3.4.3.7 virtual string EncodeAnswerCall (CSTAResponseInfo *response*, ConnectionID *callToAnswer*, string *deviceToLift*) `[protected, virtual]`

Encodes an AnswerCall message.

Parameters

response A CSTAResponseInfo object.

callToAnswer The ConnectionID of the call to answer.

deviceToLift The device (e.g., "800") that is to answer the call.

Returns

The encoded message.

3.4.3.8 virtual string EncodeAnswerCall (CSTAResponseInfo *response*, ConnectionID *callToAnswer*) `[protected, virtual]`

Encodes an AnswerCall message.

Parameters

response A CSTAResponseInfo object.

callToAnswer The ConnectionID of the call to answer.

Returns

The encoded message.

3.4.3.9 virtual string EncodeClearConnection (CSTAResponseInfo *response*, ConnectionID *connectionToClear*) `[protected, virtual]`

Encodes a ClearConnection message.

Parameters

response A CSTAResponseInfo object.

connectionToClear The ConnectionID of the connection to clear.

Returns

The encoded message.

3.4.3.10 virtual string EncodeConsultationCall (CSTAResponseInfo *response*, ConnectionID *existingCall*, string *targetDevice*) `[protected, virtual]`

Encodes a ConsultationCall message.

Parameters

response A CSTAResponseInfo object.

existingCall The connection id of the call for which the consultation call will be made.

targetDevice Identifier (e.g., phone number) of the device that is the target of the consultation call.

Returns

The encoded message.

3.4.3.11 virtual string EncodeGetAgentState (CSTAResponseInfo *response*, string *targetDevice*) `[protected, virtual]`

Encodes a GetAgentState message.

Parameters

response A CSTAResponseInfo object.

targetDevice The device whose agent state is desired.

Returns

The encoded message.

3.4.3.12 virtual string EncodeGetDND (CSTAResponseInfo *response*, string *targetDevice*) `[protected, virtual]`

Encodes a GetDoNotDisturb message.

Parameters

response A CSTAResponseInfo object.

targetDevice The phone for which the Do Not Disturb setting is desired.

Returns

The encoded message.

3.4.3.13 virtual string EncodeGetLogicalDevInfo (CSTAResponseInfo *response*, string *targetDevice*) `[protected, virtual]`

Encodes a GetLogicalDeviceInformation message.

Parameters

response A CSTAResponseInfo object.

targetDevice The device for which the information is needed.

Returns

The encoded message.

3.4.3.14 virtual string EncodeGetPhysicalDevInfo (CSTAResponseInfo *response*, string *targetDevice*) `[protected, virtual]`

Encodes a GetPhysicalDeviceInformation message.

Parameters

response A CSTAResponseInfo object.

targetDevice The device for which the information is needed.

Returns

The encoded message.

3.4.3.15 virtual string EncodeGetSFDevices (CSTAResponseInfo *response*, ReqDeviceCategory *category*) `[protected, virtual]`

Encodes a GetSwitchingFunctionDevices message.

Parameters

response A CSTAResponseInfo object.

category The category of device for which the list is desired.

Returns

The encoded message.

3.4.3.16 virtual string EncodeHoldCall (CSTAResponseInfo *response*, ConnectionID *callToHold*) `[protected, virtual]`

Encodes a HoldCall message.

Parameters

response A CSTAResponseInfo object.

callToHold The ConnectionID object for the call to put on hold.

Returns

The encoded message.

3.4.3.17 virtual string EncodeInvokeID (string *initialMessage*) `[virtual]`

This method prepends an invoke ID to an already encoded XML CSTA message.

Parameters

initialMessage The XML CSTA message without the invoke ID.

Returns

The message with the invoke ID prepended.

3.4.3.18 virtual string EncodeMakeCall (CSTAResponseInfo *response*, string *callingDevice*, string *calledDevice*) `[protected, virtual]`

Encodes a MakeCall message.

Parameters

response A CSTAResponseInfo object.

callingDevice The device that is making the call.

calledDevice The device that is being called.

Returns

The encoded message.

3.4.3.19 virtual string EncodeMonitorStart (CSTAResponseInfo *response*, ConnectionID *targetCall*) `[protected, virtual]`

Encodes a MonitorStart message to monitor a call.

Parameters

response A CSTAResponseInfo object.

targetCall The call to monitor.

Returns

The encoded message.

3.4.3.20 virtual string EncodeMonitorStart (CSTAResponseInfo *response*, string *targetDevice*) `[protected, virtual]`

Encodes a MonitorStart message to monitor a device.

Parameters

response A CSTAResponseInfo object.

targetDevice Identifier (e.g., telephone number) of the device to monitor.

Returns

The encoded message.

3.4.3.21 virtual string EncodeMonitorStop (CSTAResponseInfo *response*, string *xref*) `[protected, virtual]`

Encodes a MonitorStop message.

Parameters

response A CSTAResponseInfo object.

xref The cross reference id of the monitor request as a MonitorCrossRefID object.

Returns

The encoded message.

3.4.3.22 virtual string EncodeRequestSystemStatus (CSTAResponseInfo *response*) `[protected, virtual]`

Encodes a RequestSystemStatus message.

Parameters

response A CSTAResponseInfo object.

Returns

The encoded message.

3.4.3.23 virtual string EncodeRetrieveCall (CSTAResponseInfo *response*, ConnectionID *callToRetrieve*) `[protected, virtual]`

Encodes a RetrieveCall message.

Parameters

response A CSTAResponseInfo object.

callToRetrieve The ConnectionID of the call to retrieve.

Returns

The encoded message.

3.4.3.24 virtual string EncodeSendData (CSTAResponseInfo *response*, IOCrossRefID *xref*, string *strText*) `[protected, virtual]`

Encodes a SendData message.

Parameters

response A CSTAResponseInfo object.

xref An IOCrossRefID object, most likely obtained by a previous call to StartDataPath.

strText The text to send to the telephony device.

Returns

The encoded message.

3.4.3.25 virtual string EncodeSendStoredCDR (CSTAResponseInfo *response*, string *cdrCrossRefID*) `[protected, virtual]`

Encodes a SendStoredCallDetailRecords message.

Parameters

response A CSTAResponseInfo object.

cdrCrossRefID The CDR cross reference id that was returned in the response to a previously issued StartCDR-Transmission() call.

Returns

The encoded message.

3.4.3.26 virtual string EncodeSetAgentState (CSTAResponseInfo *response*, string *agentDevice*, ReqAgentState *agentState*) `[protected, virtual]`

Encodes a SetAgentState message.

Parameters

response A CSTAResponseInfo object.

agentDevice The device associated with the agent.

agentState An ReqAgentState object indicating the desired state of the agent.

Returns

The encoded message.

3.4.3.27 virtual string EncodeSetAgentState (CSTAResponseInfo *response*, string *agentDevice*, ReqAgentState *agentState*, string *agentID*) `[protected, virtual]`

Encodes a SetAgentState message.

Parameters

response A CSTAResponseInfo object.

agentDevice The device associated with the agent.

agentState An ReqAgentState object indicating the desired state of the agent.

agentID The agent id.

Returns

The encoded message.

3.4.3.28 virtual string EncodeSetDisplay (CSTAResponseInfo *response*, string *targetDevice*, string *text*) `[protected, virtual]`

Encodes a Set Display message.

Parameters

response A CSTAResponseInfo object.

targetDevice The device to which the text is to be sent.

text The text to be sent.

Returns

The encoded message.

3.4.3.29 virtual string EncodeSetDND (CSTAResponseInfo *response*, string *targetDevice*, bool *dndOn*) `[protected, virtual]`

Encodes a SetDoNotDisturb message.

Parameters

response A CSTAResponseInfo object.

targetDevice The device for which Do Not Disturb is to be set or cleared.

dndOn If true, indicates that Do Not Disturb is to be turned on. If false, indicates that Do Not Disturb is to be turned off.

Returns

The encoded message.

3.4.3.30 virtual string EncodeSetMsgWaiting (CSTAResponseInfo *response*, string *targetDevice*, bool *indicatorOn*) `[protected, virtual]`

Encodes a SetMessageWaiting message.

Parameters

response A CSTAResponseInfo object.

targetDevice The device for which the message waiting indicator is to be turned on or off.

indicatorOn If true, indicates that the message waiting indicator is to be turned on. If false, indicates that the message waiting indicator is to be turned off.

Returns

The encoded message.

3.4.3.31 virtual string EncodeSetRingerStatus (CSTAResponseInfo *response*, string *targetDevice*, string *targetRinger*, RingMode *rm*, long *ringPattern*) `[protected, virtual]`

Encodes a SetRingerStatus message.

Parameters

response A CSTAResponseInfo object.

targetDevice The device to ring.

targetRinger The id of the ringer to use for the ring. This argument can be specified as a character string (e.g, "abc"), a hex string (e.g, "'010A05'H"), or a binary string (e.g, "'00000001000010100000101'B").

rm A RingMode instance that indicates either ringing or notRinging.

ringPattern The indicator of the ring pattern to use.

Returns

The encoded message.

3.4.3.32 virtual string EncodeSingleStepTransfer (CSTAResponseInfo *response*, ConnectionID *callToTransfer*, string *transferToDevice*) `[protected, virtual]`

Encodes a SingleStepTransfer message.

Parameters

response A CSTAResponseInfo object.

callToTransfer A ConnectionID object that identifies the call to be transferred.

transferToDevice The device (e.g., "101") to which the call is to be transferred.

Returns

The encoded message.

3.4.3.33 virtual string EncodeSnapshotCall (CSTAResponseInfo *response*, ConnectionID *callToSnapshot*) `[protected, virtual]`

Encodes a SnapshotCall message.

Parameters

response A CSTAResponseInfo object.

callToSnapshot The ConnectionID of the call for which the snapshot is desired.

Returns

The encoded message.

3.4.3.34 virtual string EncodeSnapshotDevice (CSTAResponseInfo *response*, string *targetDevice*) `[protected, virtual]`

Encodes a SnapshotDevice message.

Parameters

response A CSTAResponseInfo object.

targetDevice Identifier (e.g., phone number) of the device for which the snapshot is desired.

Returns

The encoded message.

3.4.3.35 virtual string EncodeStartCDRTrans (CSTAResponseInfo *response*, CDRTransferMode *transferMode*) `[protected, virtual]`

Encodes a StartCallDetailRecordsTransmission message.

Parameters

response A CSTAResponseInfo object.

transferMode The mode the PBX or UA is to use to transfer call detail records.

Returns

The encoded message.

3.4.3.36 virtual string EncodeStartDataPath (CSTAResponseInfo *response*, string *targetDevice*) `[protected, virtual]`

Encodes a StartDataPath message.

Parameters

response A CSTAResponseInfo object.

targetDevice The device against which a data path is to be started.

Returns

The encoded message.

3.4.3.37 virtual string EncodeStartSession (CSTAResponseInfo *response*, string *applicationID*) `[protected, virtual]`

Encodes a StartApplicationSession message.

Parameters

response A CSTAResponseInfo object.

applicationID The application id to be encoded into the message.

Returns

The encoded message.

3.4.3.38 virtual string EncodeStopCDRTrans (CSTAResponseInfo *response*, string *cdrCrossRefID*) `[protected, virtual]`

Encodes a StopCallDetailRecordsTransmission message.

Parameters

response A CSTAResponseInfo object.

cdrCrossRefID The CDR cross reference id that was returned in the response to a previously issued StartCDR-Transmission() call.

Returns

The encoded message.

3.4.3.39 virtual string EncodeStopDataPath (CSTAResponseInfo *response*, IOCrossRefID *xref*) `[protected, virtual]`

Encodes a StopDataPath message.

Parameters

response A CSTAResponseInfo object.

xref An IOCrossRefID object, most likely obtained from a previous call to StartDataPath.

Returns

The encoded message.

3.4.3.40 virtual string EncodeStopSession (CSTAResponseInfo *response*) `[protected, virtual]`

Encodes a StopApplicationSession message.

Parameters

response A CSTAResponseInfo object.

Returns

The encoded message.

3.4.3.41 virtual string EncodeTransferCall (CSTAResponseInfo *response*, ConnectionID *initiatedCall*, ConnectionID *originalCall*) `[protected, virtual]`

Encodes a TransferCall message.

Parameters

response A CSTAResponseInfo object.

initiatedCall ConnectionID of the new call initiated by the consultation call. The initiatedCall member of the ConsultationCallResponse_ELEM class, for example, contains this ConnectionID.

originalCall ConnectionID of the original call. The somewhat confusingly named callingDevice member of the MakeCallResponse_ELEM class contains this ConnectionID, as does the establishedConnection member of the EstablishedEvent_ELEM class.

Returns

The encoded message.

3.4.3.42 virtual CSTAResponseInfo GetAgentState (string *agentDevice*) `[virtual]`

Gets the state of the agent associated with a device.

Parameters

agentDevice The device associated with the agent.

Returns

A CSTAResponseInfo object.

3.4.3.43 virtual CSTAResponseInfo GetDoNotDisturb (string *targetDevice*) `[virtual]`

Gets the Do Not Disturb setting for a phone.

Parameters

targetDevice The phone for which the Do Not Disturb setting is desired.

Returns

A CSTAResponseInfo object.

3.4.3.44 virtual CSTAResponseInfo GetLogicalDevInfo (string *targetDevice*) `[virtual]`

Gets information about the logical element of a device.

Parameters

targetDevice The device for which the information is desired.

Returns

A CSTAResponseInfo object.

3.4.3.45 virtual CSTAResponseInfo GetPhysicalDevInfo (string *targetDevice*) `[virtual]`

Gets information about the physical element of a device.

Parameters

targetDevice The device for which the information is desired.

Returns

A CSTAResponseInfo object.

3.4.3.46 virtual CSTAResponseInfo GetSFDevices () `[virtual]`

Sends a Get Switching Function Devices request to the PBX or UA.

Returns

A CSTAResponseInfo object.

3.4.3.47 virtual CSTAResponseInfo GetSFDevices (ReqDeviceCategory *deviceCategory*) `[virtual]`

Sends a Get Switching Function Devices request to the PBX or UA.

Parameters

deviceCategory The category of device for which the list is desired.

Returns

A CSTAResponseInfo object.

3.4.3.48 virtual CSTAResponseInfo HoldCall (ConnectionID *callToHold*) `[virtual]`

Instruct the PBX or UA to hold a call.

Parameters

callToHold The ConnectionID of the call to be held.

Returns

A CSTAResponseInfo object.

3.4.3.49 virtual CSTAResponseInfo MakeCall (string *callingDevice*, string *calledDevice*) `[virtual]`

Instruct the PBX or UA to place a call.

Parameters

callingDevice Identifier (e.g., phone number) of the device making the call.

calledDevice Identifier (e.g., phone number) of the device being called.

Returns

A CSTAResponseInfo object.

3.4.3.50 virtual CSTAResponseInfo MonitorStart (ConnectionID *callToMonitor*) `[virtual]`

Issues a MonitorStart request to the PBX or UA to monitor a call.

Parameters

callToMonitor The call to monitor.

Returns

A CSTAResponseInfo object.

3.4.3.51 virtual CSTAResponseInfo MonitorStart (string *deviceToMonitor*) `[virtual]`

Issues a MonitorStart request to the PBX or UA to monitor a device.

Parameters

deviceToMonitor Identifier (e.g., telephone number) of the device to monitor.

Returns

A CSTAResponseInfo object.

3.4.3.52 virtual CSTAResponseInfo MonitorStop (string *crossRefID*) `[virtual]`

Stop a previously started PBX or UA monitor request.

Parameters

crossRefID The cross reference id of the monitor request as a MonitorCrossRefID object.

Returns

A CSTAResponseInfo object.

3.4.3.53 virtual CSTAResponseInfo MonitorStopAtDevice (string *monitoredDevice*) `[virtual]`

This method stops all monitors active against the indicated device, regardless of what thread started the monitor. The method will only stop monitors started through the MonitorStart() method.

Parameters

monitoredDevice The monitored device (e.g., extension).

Returns

If no problems are encountered, the method returns a CSTAResponseInfo object containing the response from the PBX or UA for the LAST MonitorStop message.
If any problems are encountered, the method returns a CSTAResponseInfo object containing information about the error, including any response from the PBX for the problematic MonitorStop message.

3.4.3.54 virtual CSTAResponseInfo RequestSystemStatus () `[virtual]`

Retrieves a system status from the PBX or UA.

Returns

A CSTAResponseInfo object.

3.4.3.55 virtual CSTAResponseInfo RetrieveCall (ConnectionID *callToRetrieve*) `[virtual]`

Retrieves a held call.

Parameters

callToRetrieve The ConnectionID of the call to retrieve.

Returns

A CSTAResponseInfo object.

3.4.3.56 virtual CSTAResponseInfo RingDevice (string *targetDevice*, string *targetRinger*, long *ringPattern*) `[virtual]`

Causes a telephony device to ring.

Parameters

targetDevice The device to ring.

targetRinger The id of the ringer to use for the ring. This argument can be specified as a character string (e.g, "abc"), a hex string (e.g, "'010A05'H"), or a binary string (e.g, "'00000001000010100000101'B").

ringPattern The indicator of the ring pattern to use.

Returns

A CSTAResponseInfo object.

3.4.3.57 virtual CSTAResponseInfo SendData (IOCrossRefID *xref*, string *text*) `[virtual]`

Sends a text message to a telephony device.

Parameters

xref An IOCrossRefID object, most likely obtained by a previous call to StartDataPath.

text The text to send to the telephony device.

Returns

A CSTAResponseInfo object.

3.4.3.58 virtual CSTAResponseInfo SendStoredCDR (string *cdrCrossRefID*) `[virtual]`

Issues a SendStoredCallDetailRecords request to the PBX or UA. A CDR callback method (see PBXSession.XMLCDRCallback) must be defined in order to receive CDR messages.

Parameters

cdrCrossRefID The CDR cross reference id that was returned in the response to a previously issued StartCDRTransmission() call.

Returns

A CSTAResponseInfo object.

3.4.3.59 virtual CSTAResponseInfo SetAgentState (string *agentDevice*, ReqAgentState *agentState*) `[virtual]`

Sets the state of an agent associated with a device.

Parameters

agentDevice The device associated with the agent.

agentState The desired state for the agent.

Returns

A CSTAResponseInfo object.

3.4.3.60 virtual CSTAResponseInfo SetAgentState (string *agentDevice*, ReqAgentState *agentState*, string *agentID*) `[virtual]`

Sets the state of an agent associated with a device.

Parameters

agentDevice The device associated with the agent.

agentState The desired state for the agent.

agentID The agent id.

Returns

A CSTAResponseInfo object.

3.4.3.61 virtual CSTAResponseInfo SetDisplay (string *targetDevice*, string *text*) `[virtual]`

Sends text to a telephony device's display

Parameters

targetDevice The device to which the text is to be sent.

text The text to be sent.

Returns

A CSTAResponseInfo object.

3.4.3.62 virtual CSTAResponseInfo SetDoNotDisturb (string *targetDevice*) `[virtual]`

Sets the Do Not Disturb feature for a phone.

Parameters

targetDevice The device for which Do Not Disturb is to be set.

Returns

A CSTAResponseInfo object.

3.4.3.63 virtual CSTAResponseInfo SetMessageWaiting (string *targetDevice*) `[virtual]`

Turns on the message waiting indicator on a device's display.

Parameters

targetDevice The device for which the indicator is to be turned on.

Returns

A CSTAResponseInfo object.

3.4.3.64 virtual CSTAResponseInfo SingleStepTransfer (ConnectionID *callToTransfer*, string *transferToDevice*) `[virtual]`

Perform a single step transfer.

Parameters

callToTransfer A ConnectionID object that identifies the call to be transferred.

transferToDevice The device (e.g., "101") to which the call is to be transferred.

Returns

A CSTAResponseInfo object.

3.4.3.65 virtual CSTAResponseInfo SnapshotCall (ConnectionID *callToSnapshot*) `[virtual]`

Instruct the PBX or UA to take a snapshot of a call.

Parameters

callToSnapshot The ConnectionID of the call for which the snapshot is desired.

Returns

A CSTAResponseInfo object.

3.4.3.66 virtual CSTAResponseInfo SnapshotDevice (string *deviceToSnapshot*) `[virtual]`

Instruct the PBX to take a snapshot of calls active at a device.

Parameters

deviceToSnapshot Identifier (e.g., phone number) of the device for which the snapshot is desired.

Returns

A CSTAResponseInfo object.

3.4.3.67 virtual CSTAResponseInfo StartCDRTransmission (CDRTransferMode *transferMode*) `[virtual]`

Issues a StartCallDetailRecordsTransmission request to the PBX or UA. A CDR callback method (see PBXSession.XMLCDRCallback) must be defined in order to receive CDR messages.

Parameters

transferMode Indicates how the PBX is to transfer the CDR information.

Returns

A CSTAResponseInfo object.

3.4.3.68 virtual CSTAResponseInfo StartDataPath (string *targetDevice*) `[virtual]`

Opens up a data path to a specified device.

Parameters

targetDevice Specifies the device to which a data path is to be opened.

Returns

A CSTAResponseInfo object.

3.4.3.69 virtual CSTAResponseInfo StartSession (string *applicationID*) `[virtual]`

Establish a session with the PBX.

Parameters

applicationID A free text string to identify the application.

Returns

A CSTAResponseInfo object.

Reimplemented in uaXMLed4.

3.4.3.70 virtual CSTAResponseInfo StartSession () `[virtual]`

Establish a session with the PBX, using "CSTADLL" as the application identifier.

Returns

A CSTAResponseInfo object.

Reimplemented in uaXMLed4.

3.4.3.71 virtual CSTAResponseInfo StopCDRTransmission (string *cdrCrossRefID*) `[virtual]`

Issues a StopCallDetailRecordsTransmission request to the PBX or UA.

Parameters

cdrCrossRefID The CDR cross reference id that was returned in the response to a previously issued StartCDR-Transmission() call.

Returns

A CSTAResponseInfo object.

3.4.3.72 virtual CSTAResponseInfo StopDataPath (IOCrossRefID *xref*) `[virtual]`

Stops a previously established data path

Parameters

xref An IOCrossRefID object, most likely obtained from a previous call to StartDataPath.

Returns

A CSTAResponseInfo object.

3.4.3.73 virtual CSTAResponseInfo StopRing (string *targetDevice*, string *targetRinger*, long *ringPattern*) `[virtual]`

Stops a ringer on a telephony device.

Parameters

targetDevice The device for which the ringer is to stop.

targetRinger The id of the ringer to stop. This argument can be specified as a character string (e.g, "abc"), a hex string (e.g, "'010A05'H"), or a binary string (e.g, "'00000001000010100000101'B").

ringPattern The indicator of the ring pattern to stop.

Returns

A CSTAResponseInfo object.

3.4.3.74 virtual CSTAResponseInfo StopSession () `[virtual]`

Stops a session with a PBX. The TCP/IP connection to the PBX will be terminated.

Returns

A CSTAResponseInfo object.

Reimplemented in uaXMLed4.

3.4.3.75 virtual CSTAResponseInfo TransferCall (ConnectionID *initiatedCall*, ConnectionID *originalCall*) `[virtual]`

Transfers a call. A consultation call must be done before calling this method.

Parameters

initiatedCall ConnectionID of the new call initiated by the consultation call. The initiatedCall member of the ConsultationCallResponse_ELEM class, for example, contains this ConnectionID.

originalCall ConnectionID of the original call. The somewhat confusingly named callingDevice member of the MakeCallResponse_ELEM class contains this ConnectionID, as does the establishedConnection member of the EstablishedEvent_ELEM class.

Returns

A CSTAResponseInfo object.

3.4.4 Property Documentation

3.4.4.1 PBXSession SessionObject `[get]`

The PBXSession object associated with this instance.

3.4.4.2 CSTAContext ThreadContext `[get]`

The CSTAContext structure for this thread.

3.5 LicenseException Class Reference

3.5.1 Detailed Description

Defines an exception that occurs while trying to find license information.

3.6 LicenseOptions Class Reference

3.6.1 Detailed Description

This class holds booleans that define what capabilities are defined in the license.

3.7 PBXSession Class Reference

Public Member Functions

- delegate void AsyncCallback (PBXSession sessionObject, byte[ ] asyncData)
- delegate void AsyncExceptionCallback (PBXSession sessionObject, ApplicationException exception)
- void Close (CSTAContext threadContext)
- delegate void ConnectionCallback (PBXSession sessionObject)
- void Open (CSTAContext threadContext)
- PBXSession (string pbxSystem, int port)
- SocketState SendACSEMessage (byte[ ] message, int messageLength, Constants.ACSEMessageTypes messageType, CSTAContext threadContext)
- void SendMessage (string messageType, byte[ ] message, int messageLength, CSTAContext threadContext)
- void SendMessage (byte[ ] message, int messageLength, CSTAContext threadContext)
- void SendXMLMessage (string messageType, string strMessage, CSTAContext threadContext)
- void SendXMLMessage (string strMessage, CSTAContext threadContext)
- SocketState SendXMLSession (string strMessage, Constants.XMLSessionMessageTypes enmMessageType, CSTAContext threadContext)
- void WaitForROSEResponse (CSTAContext threadContext)
- void WaitForXMLResponse (CSTAContext threadContext)
- delegate void XMLAsyncCallback (PBXSession sessionObject, string message)

Properties

- Constants.CallbackInvocationMechanisms CallbackInvocationMechanism `[get, set]`
- AsyncCallback CDRCallback `[get, set]`
- AsyncCallback ClientCallback `[get, set]`
- bool Connected `[get, set]`
- ConnectionCallback ConnectionLostCallback `[get, set]`
- bool DebugMode `[get, set]`
- AsyncExceptionCallback ExceptionCallback `[get, set]`
- int MaxReceiveTimeout `[get, set]`
- Constants.Encoding MessageEncoding `[get, set]`
- string PBXSystem `[get]`
- int Port `[get]`
- AsyncCallback SystemStatusCallback `[get, set]`
- XMLAsyncCallback XMLCDRCallback `[get, set]`
- XMLAsyncCallback XMLClientCallback `[get, set]`
- XMLAsyncCallback XMLSystemStatusCallback `[get, set]`

3.7.1 Detailed Description

This class manages communication with a PBX. One instance of this class should be created for each PBX with which a CSTADLL client application needs to exchange CSTA messages.

The CSTA worker classes (e.g., Alcatel4400, PanasonicNCP) hold a reference to a PBXSession object. If the constructor for the worker class that takes a PBX identification and a PBX port is used, a PBXSession object is created. Alternatively, the client application can create a PBXSession instance and pass a reference to the instance to the other worker class constructor signature.

Only one PBXSession instance for a PBX/port combination should be created. The behavior is undefined if multiple PBXSession instances are created for the same PBX and port.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 PBXSession (string *pbxSystem*, int *port*)

Constructs a PBXSession object.

Parameters

pbxSystem The name or IP address of the PBX system.

port The port on the PBX system to which the client is connecting.

3.7.3 Member Function Documentation

3.7.3.1 delegate void AsyncCallback (PBXSession *sessionObject*, byte[ ] *asyncData*)

Declaration of a callback function to be invoked when one of the following messages is received: a monitor event report message, a route message, a CDR Report message, a CDR Notification message, or a system status request.

Parameters

sessionObject The session object for the PBX that generated the asynchronous message.

asyncData The data received asynchronously from the PBX.

3.7.3.2 delegate void AsyncExceptionCallback (PBXSession *sessionObject*, ApplicationException *exception*)

Declaration of a callback function to be invoked if a condition is encountered in the asynchronous I/O handler that would otherwise result in an exception being thrown. Note that in a couple of cases the asynchronous code will still throw an exception, even if this callback is defined.

Parameters

sessionObject The session object for the PBX that sent a packet that triggered an exception condition.

exception The ApplicationException object that would have been thrown in the asynchronous I/O handling code if this callback were not defined.

3.7.3.3 void Close (CSTAContext *threadContext*)

Terminates the session to the PBX. This method can be used to terminate sessions with PBX devices that don't accept ACSE release association requests.

Parameters

threadContext The context object for the caling thread.

3.7.3.4 delegate void ConnectionCallback (PBXSession *sessionObject*)

Declaration of a callback function to be invoked if the connection to the PBX is lost.

Parameters

sessionObject The session object for the PBX whose connection was lost.

3.7.3.5 void Open (CSTAContext *threadContext*)

This method can be used to establish communication with a PBX device before any messages are actually sent to the device. TCP/IP connectivity is established and an asynchronous read is started to receive messages sent from the PBX.

Parameters

threadContext The thread context object.

3.7.3.6 SocketState SendACSEMessage (byte[ ] *message*, int *messageLength*, Constants.ACSEMessageTypes *messageType*, CSTAContext *threadContext*)

This method sends an ACSE message (either Make Association or Release Association) to the PBX and receives the response. This operation is done synchronously. If the Make Association needs to be done (usually it does), it must be done before any threads for sending and receiving CSTA messages are started.

This method is only intended to be used by client code that encodes its own ACSEMakeAssociation or ACSEReleaseAssociation message. Most clients can probably use the MakeACSEAssociation() and ReleaseACSEAssociation() methods that are in each phase's helper classes.

Parameters

message An encoded ACSE Make Association or Release Association message.

messageLength The length of the encoded message.

messageType A constant telling whether the message is an ACSE Make Association or an ACSE Release Association.

threadContext The thread context object.

Returns

A populated SocketState instance.

3.7.3.7 void SendMessage (string *messageType*, byte[ ] *message*, int *messageLength*, CSTAContext *threadContext*)

This method sends a message to the PBX using TCP/IP.

Parameters

messageType A string token to help identify the message in the CSTADLL log file.

message Byte array containing the encoded message to send.

messageLength The length of the encoded message.

threadContext The thread context object.

3.7.3.8 void SendMessage (byte[ ] *message*, int *messageLength*, CSTAContext *threadContext*)

This method sends a message to the PBX using TCP/IP.

Parameters

message Byte array containing the encoded message to send.

messageLength The length of the encoded message.

threadContext The thread context object.

3.7.3.9 void SendXMLMessage (string *messageType*, string *strMessage*, CSTAContext *threadContext*)

This method sends an XML message to the PBX using TCP/IP.

Parameters

messageType A string token to help identify the message in the CSTADLL log file.

strMessage The XML message to send.

threadContext The thread context object.

3.7.3.10 void SendXMLMessage (string *strMessage*, CSTAContext *threadContext*)

This method sends an XML message to the PBX using TCP/IP.

Parameters

strMessage The XML message to send.

threadContext The thread context object.

3.7.3.11 SocketState SendXMLSession (string *strMessage*, Constants.XMLSessionMessageTypes *enmMessageType*, CSTAContext *threadContext*)

This method sends an XML session management (ECMA-354) message to the PBX.

Parameters

strMessage The text of the XML message to send.

enmMessageType A constant indicating what kind of session management message is being sent.

threadContext The thread context object.

Returns

A populated SocketState instance if the message is a StartSession message. Null if the message is StopSession or ResetSession.

3.7.3.12 void WaitForROSEResponse (CSTAContext *threadContext*)

This method waits for a response to a CSTA message sent with a ROSE header.

Parameters

threadContext The CSTAContext object associated with the calling thread.

3.7.3.13 void WaitForXMLResponse (CSTAContext *threadContext*)

This method waits for a response to an XML CSTA message.

Parameters

threadContext The CSTAContext object associated with the calling thread.

3.7.3.14 delegate void XMLAsyncCallback (PBXSession *sessionObject*, string *message*)

Declaration of a callback function to be invoked when an asynchronous XML monitor event or route messae is received.

Parameters

sessionObject The session object for the PBX or UA that generated the asynchronous message.

message The text of the message received asynchronously from the PBX or UA.

3.7.4 Property Documentation

3.7.4.1 Constants.CallbackInvocationMechanisms CallbackInvocationMechanism `[get, set]`

Indicates what asynchronous callback invocation mechanism to use. See the documentation on Constants.CallbackInvocationMechanisms for a detailed description of the possible options.

3.7.4.2 AsyncCallback CDRCallback `[get, set]`

Holds a reference to an asynchronous callback function that will be invoked when CDR information is received asynchronously. This information could take the form of a CDR Report message or a CDR Notification message. If the latter, the application should use the SendStoredCDR() method to request the stored CDRs from the PBX.

3.7.4.3 AsyncCallback ClientCallback `[get, set]`

Holds a reference to an asynchronous callback function. This function will be invoked if a monitor event or route message is received asynchronously from the PBX.

3.7.4.4 bool Connected `[get, set]`

Indicates whether the session to the PBX is connected.

3.7.4.5 ConnectionCallback ConnectionLostCallback `[get, set]`

Holds a reference to an asynchronous callback function. This function will be invoked if the connection to the PBX is lost.

3.7.4.6 bool DebugMode `[get, set]`

Enables behavior that facilitates debugging of the CSTADLL software. This property is most likely useful only to Objective Systems staff.

3.7.4.7 AsyncExceptionCallback ExceptionCallback `[get, set]`

Holds a reference to an asynchronous callback function. This function will be invoked if a condition occurs in the asynchronous I/O handler that otherwise would have resulted in an exception being thrown if this callback were not defined. Note that in a couple of cases the asynchronous code will still throw an exception, even if this callback is defined.

3.7.4.8 int MaxReceiveTimeout `[get, set]`

Specifies the amount of time, in milliseconds, to wait for a respone to arrive from the PBX. The default value is 5,000 milliseconds (5 seconds).

3.7.4.9 Constants.Encoding MessageEncoding `[get, set]`

Indicates how messages exchanges with this PBX are encoded.

3.7.4.10 string PBXSystem `[get]`

The TCIP/IP address or well-known name of the PBX.

3.7.4.11 int Port `[get]`

The port where the PBX listens for CSTA messages.

3.7.4.12 AsyncCallback SystemStatusCallback `[get, set]`

Holds a reference to an asynchronous callback function. This function will be invoked if a BER System Status message is received asynchronously from the PBX. The client application does NOT need to send a System Status Response, since that is taken care of by CSTADLL.

3.7.4.13 XMLAsyncCallback XMLCDRCallback `[get, set]`

Holds a reference to an asynchronous callback function that will be invoked when XML CDR information is received asynchronously.

This information could take the form of a CDR Report message or a CDR Notification message. If the latter, the application should use the SendStoredCDR() method to request the stored CDRs from the PBX or UA.

3.7.4.14 XMLAsyncCallback XMLClientCallback `[get, set]`

Holds a reference to an asynchronous XML callback function. This function will be invoked if an XML monitor event report message or route message is received asynchronously from the PBX or UA.

3.7.4.15 XMLAsyncCallback XMLSystemStatusCallback `[get, set]`

Holds a reference to an asynchronous callback function. This function will be invoked if an XML System Status message is received asynchronously from the PBX. The client application does NOT need to send a System Status Response, since that is taken care of by CSTADLL.

3.8 PBXSessionException Class Reference

3.8.1 Detailed Description

Defines an exception that occurs while communicating with a PBX.

3.9 PBXSessionHelper Class Reference

Properties

- static bool LoggingEnabled `[get, set]`
- static string LoggingFolder `[get, set]`
- static long MaxLogFileSize `[get, set]`

3.9.1 Detailed Description

This class holds static properties that affect all PBX sessions.

3.9.2 Property Documentation

3.9.2.1 bool LoggingEnabled `[static, get, set]`

Indicates whether logging should be done.

3.9.2.2 string LoggingFolder `[static, get, set]`

Specifies a folder to receive the log file. If not specified, the log file will go into whatever folder the calling .exe resides in.

3.9.2.3 long MaxLogFileSize `[static, get, set]`

Defines the maximum size, in bytes, that a log file is allowed to grow to before a new log file is opened. If no value is specified for this property, the maximum size is Constants.MAX_LOG_FILE_SIZE. Any value specified for this property overrides this default setting.

3.10 PBXSessionHelperEd4 Class Reference

3.10.1 Detailed Description

This class contains utility methods used by PBXSessionHelper for XML edition 4.

3.11 ResetSessionInfo Class Reference

3.11.1 Detailed Description

This class provides information that needs to be passed to the thread that periodically sends an XML ResetSession message to the PBX.

3.12 SocketState Class Reference

Properties

- byte[ ] AckBuffer `[get, set]`
- byte[ ] ReadBuffer `[get, set]`
- List< byte[ ]> ReadBuffers `[get, set]`
- int TotalLength `[get, set]`

3.12.1 Detailed Description

This class contains the response received from the PBX and state information about the exchange with the PBX that is used internally by CSTADLL.

3.12.2 Property Documentation

3.12.2.1 byte [ ] AckBuffer `[get, set]`

Contains the first response from the PBX for situations where the PBX sends multiple response messages (e.g., Get Switching Function Devices). The data messages that are sent after this ack will be in ReadBuffers.

3.12.2.2 byte [ ] ReadBuffer `[get, set]`

Contains the bytes most recently read from the socket. This buffer will be filled in bit by bit as the message is read.

3.12.2.3 List<byte[ ]> ReadBuffers `[get, set]`

Contains multiple collections of bytes read from the socket. This array is used for situations where a response to a message comes in multiple segments (e.g., Get Switching Function Devices). For these situations the immediate response will be in AckBuffer.

3.12.2.4 int TotalLength `[get, set]`

The total length of a complete message received from the PBX. This is also used as an offset into the read buffer so we can build the message as it's received.

3.13 uaSIPInvite Class Reference

Properties

- string CallId `[get, set]`
- string Contact `[get, set]`
- string From `[get, set]`
- string InviteTarget `[get, set]`
- ushort MaxForwards `[get, set]`
- string Via `[get, set]`

3.13.1 Detailed Description

This class holds information needed in order to do a SIP INVITE to a UA for a uaCSTA session.

3.13.2 Property Documentation

3.13.2.1 string CallId `[get, set]`

Contains the value for the Call-ID: header in the SIP INVITE header block.

3.13.2.2 string Contact `[get, set]`

Contains the value for the Contact: header in the SIP INVITE header block.

3.13.2.3 string From `[get, set]`

Contains the value for the From: header in the SIP INVITE header block.

3.13.2.4 string InviteTarget `[get, set]`

Contains the identifier of the UA with which to initiate a uaCSTA session via a SIP INVITE message.

3.13.2.5 ushort MaxForwards `[get, set]`

Contains the value for the Max-Forwards: header in the SIP INVITE header block. Per RFC 3261, the default value is 70.

3.13.2.6 string Via `[get, set]`

Contains the value for the Via: header in the SIP INVITE header block.

3.14 uaXMLed4 Class Reference

Inherits Com::Objsys::Csta::Xmled4::GenericXMLed4.

Public Member Functions

- virtual void Bye ()
- virtual CSTAResponseInfo Invite (uaSIPInvite inviteObject)
- sealed override CSTAResponseInfo StartSession ()
- sealed override CSTAResponseInfo StartSession (string applicationID)
- sealed override CSTAResponseInfo StopSession ()
- uaXMLed4 (PBXSession sessionObject)
- uaXMLed4 (string ua, int port)

3.14.1 Detailed Description

Implements uaCSTA phase 3 operations using XML edition 4. Note that most PBXes don't support all CSTA messages, so some methods in this class may result in an error status being returned by your PBX.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 uaXMLed4 (string *ua*, int *port*)

Constructs an instance associated with the given UA identifier and port.

Parameters

ua Well-known name or IP address of the SIP user agent.

port Port on which the UA listens for uaCSTA messages.

3.14.2.2 uaXMLed4 (PBXSession *sessionObject*)

Constructs an instance associated with the given PBXSession object.

Parameters

sessionObject A PBXSession object.

3.14.3 Member Function Documentation

3.14.3.1 virtual void Bye () `[virtual]`

Sends a SIP BYE message to a UA and closes TCP/IP communication with the UA.

3.14.3.2 virtual CSTAResponseInfo Invite (uaSIPInvite *inviteObject*) `[virtual]`

Sends a SIP INVITE message to a User Agent.

Parameters

inviteObject A uaSIPInvite object.

Returns

A CSTAResponseInfo object.

3.14.3.3 sealed override CSTAResponseInfo StartSession () `[virtual]`

For regular XML CSTA, the StartSession() method starts communication with a PBX. But for uaCSTA with SIP, its use is not valid. The Invite() method must be used instead. This method will throw an exception.

Returns

Nothing. An exception is thrown.

Reimplemented from GenericXMLed4.

3.14.3.4 sealed override CSTAResponseInfo StartSession (string *applicationID*) `[virtual]`

For regular XML CSTA, the StartSession() method starts communication with a PBX. But for uaCSTA with SIP, its use is not valid. The Invite() method must be used instead. This method will throw an exception.

Parameters

applicationID Ignored.

Returns

Nothing. An exception is thrown.

Reimplemented from GenericXMLed4.

3.14.3.5 sealed override CSTAResponseInfo StopSession () `[virtual]`

For regular XML CSTA, the Stp[Session() method stops communication with a PBX. But for uaCSTA with SIP, its use is not valid. The Bye() method must be used instead. This method will throw an exception.

Returns

Reimplemented from GenericXMLed4.

3.15 UnifyOpenscapeVoice Class Reference

Inherits Com::Objsys::Csta::Xmled4::GenericXMLed4.

Public Member Functions

- UnifyOpenscapeVoice (PBXSession sessionObject)
- UnifyOpenscapeVoice (string pbxSystem, int port)

3.15.1 Detailed Description

Implements CSTA XML operations for the Unify Openscape Voice PBX device.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 UnifyOpenscapeVoice (string *pbxSystem*, int *port*)

Constructs an instance associated with the given PBX identifier and port.

Parameters

pbxSystem Well-known name or IP address of the PBX.

port Port on which the PBX listens for CSTA messages.

3.15.2.2 UnifyOpenscapeVoice (PBXSession *sessionObject*)

Constructs an instance associated with the given PBXSession object.

Parameters

sessionObject A PBXSession object.

Index