packetvideo

OpenCORE™

PV2Way Engine

Build Version: CORE_9.000.1.1_RC2

May 7, 2010

Contents

Chapter 1

Data Structure Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 CPV2WayEngineFactory Class Reference

```
#include <pv_2way_engine_factory.h>
```

Static Public Member Functions

- static OSCL_IMPORT_REF void Init ()
- static OSCL_IMPORT_REF void Cleanup ()
- static OSCL_IMPORT_REF CPV2WayInterface ∗ CreateTerminal (PV2WayTerminalType aTerminalType, PVCommandStatusObserver ∗aCmdStatusObserver, PVInformationalEventObserver ∗aInfoEventObserver, PVErrorEventObserver ∗aErrorEventObserver)
- static OSCL_IMPORT_REF void DeleteTerminal (CPV2WayInterface ∗terminal)

4.1.1 Member Function Documentation

4.1.1.1 static OSCL_IMPORT_REF void CPV2WayEngineFactory::Cleanup () `[static]`

4.1.1.2 static OSCL_IMPORT_REF CPV2WayInterface∗ CPV2WayEngineFactory::CreateTerminal (PV2WayTerminalType *aTerminalType*, PVCommandStatusObserver ∗ *aCmdStatusObserver*, PVInformationalEventObserver ∗ *aInfoEventObserver*, PVErrorEventObserver ∗ *aErrorEventObserver*) `[static]`

Creates an instance of a terminal of a particular type. Initially, this will support 324m type terminals.

Parameters

aTerminalType	the type of terminal to be created.
aCmdStatusObserver	the observer for command status
aInfoEventObserver	the observer for unsolicited informational events
aErrorEventObserver	the observer for unsolicited error events

Returns

A pointer to a terminal or leaves if the type is invalid or the system is out of resources

4.1.1.3 static OSCL_IMPORT_REF void CPV2WayEngineFactory::DeleteTerminal (CPV2WayInterface ∗ *terminal*) `[static]`

This function allows the application to delete an instance of a terminal and reclaim all allocated resources. A terminal should be deleted only in the EIdle state. An attempt to delete a terminal in any other state will result in unpredictable behavior.

Parameters

terminal	the terminal to be deleted.

4.1.1.4 static OSCL_IMPORT_REF void CPV2WayEngineFactory::Init () `[static]`

The documentation for this class was generated from the following file:

- pv_2way_engine_factory.h

4.2 CPV2WayInterface Class Reference

```
#include <pv_2way_interface.h>
```

Public Member Functions

- virtual ~CPV2WayInterface ()
- virtual OSCL_IMPORT_REF PVCommandId GetSDKInfo (PVSDKInfo &aSDKInfo, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId GetSDKModuleInfo (PVSDKModuleInfo &aSDKModuleInfo, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId Init (PV2WayInitInfo &aInitInfo, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId Reset (OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId AddDataSource (PVTrackId aTrackId, PVMFNodeInterface &aDataSource, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId RemoveDataSource (PVMFNodeInterface &aDataSource, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId AddDataSink (PVTrackId aTrackId, PVMFNodeInterface &aDataSink, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId RemoveDataSink (PVMFNodeInterface &aDataSink, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId Connect (const PV2WayConnectOptions &aOptions, PVMFNodeInterface ∗aCommServer=NULL, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId Disconnect (OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId GetState (PV2WayState &aState, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId Pause (PV2WayDirection aDirection, PVTrackId aTrackId, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId Resume (PV2WayDirection aDirection, PVTrackId aTrackId, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId SetLogAppender (const char ∗aTag, OsclSharedPtr< PVLoggerAppender > &aAppender, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId RemoveLogAppender (const char ∗aTag, OsclSharedPtr< PVLoggerAppender > &aAppender, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId SetLogLevel (const char ∗aTag, int32 aLevel, bool aSetSubtree=false, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId GetLogLevel (const char ∗aTag, int32 &aLogInfo, OsclAny ∗aContextData=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId QueryInterface (const PVUuid &aUuid, PVInterface ∗&aInterfacePtr, OsclAny ∗aContext=NULL)=0
- virtual OSCL_IMPORT_REF PVCommandId CancelAllCommands (OsclAny ∗aContextData=NULL)=0

4.2.1 Detailed Description

CPV2WayInterface Class

CPV2WayInterface is the interface to the pv2way SDK, which allows initialization, control, and termination of a two-way (3g-324m, SIP) terminal. The application is expected to contain and maintain a pointer to the CPV2WayInterface instance at all times that a call is active. The CPV2WayFactory factory class is to be used to create and delete instances of this class

4.2.2 Constructor & Destructor Documentation

4.2.2.1 virtual CPV2WayInterface::∼CPV2WayInterface () `[inline, virtual]`

Object destructor function Releases Resources prior to destruction

4.2.3 Member Function Documentation

4.2.3.1 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::AddDataSink (PVTrackId *aTrackId*, PVMFNodeInterface & *aDataSink*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This function allows the user to specify the media sink for an incoming track. AddDataSinkL can be called only for established incoming tracks identified by a unique track id. Incoming tracks are initiated by the peer and their establishment is indicated using the PVT_INDICATION_INCOMING_TRACK notification which provides the media type and a unique track id. The format type is indicated using the PV2WayTrackInfoInterface extension interface in the PVAsyncInformationalEvent. Data sinks could be of the following types: a)raw media sinks like video display sinks for RGB and YUV formats, audio rendering sinks for PCM. b)sources of compressed data like file, combined decode and render devices.

Parameters

aTrackId Indicates the unique track id to be associated with this sink.

aDataSink The data sink to be added

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.2.3.2 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::AddDataSource (PVTrackId *aTrackId*, PVMFNodeInterface & *aDataSource*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This function allows the user to specify the media source for an outgoing track. Sources should be added after the PVT_INDICATION_OUTGOING_TRACK is received which specifies the format type and the unique track id. The format type is indicated using the PV2WayTrackInfoInterface extension interface in the PVAsyncInformationalEvent. Data sources could be of the following types: a)raw media sources like camera, microphone etc. b)sources of compressed data like file, combined capture and encode devices.

Parameters

aTrackId The outgoing track id

aDataSource Reference to the data source for this track

aContextData Optional opaque data that will be passed back to the user with the command response This method can leave with one of the following error codes PVMFErrNotSupported if the format of the sources/sinks is incomtible with what the SDK can handle KPVErrInvalidState if invoked in the incorrect state KErrNoMemory if the SDK failed to allocate memory during this operation

Returns

A unique command id for asynchronous completion

4.2.3.3 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::CancelAllCommands (OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API is to allow the user to cancel all pending requests. The current request being processed, if any, will also be aborted.

Parameters

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.2.3.4 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::Connect (const PV2WayConnectOptions & *aOptions*, PVMFNodeInterface ∗ *aCommServer* = `NULL`, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This function can be invoked only in the ESetup state. The terminal starts connecting with the remote terminal based on the specified options and capabilities. Incoming tracks may be opened before ConnectL completes and will be indicated via the PVT_INDICATION_INCOMING_TRACK event.

Parameters

aOptions Optional additional information for call setup.

aCommServer An optional pointer to a comm server to provide comm source and sink end-points.

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.2.3.5 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::Disconnect (OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

The Disconnect call is valid only when invoked in the EConnecting, and EConnected states. It causes the terminal to transition to the EDisconnecting state. All the media tracks both incoming and outgoing will be closed on invoking Disconnect. On completion, the terminal goes to the ESetup state. The statistics of the previous call shall still be available until Connect is invoked again.

It is a no-op when called in any other state.

The post disconnect option specifies what this terminal wishes to do after the data call is terminated, whether it wants to disconnect the line or continue the call as a voice only call.

This is an asynchronous request.

Parameters

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.2.3.6 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::GetLogLevel (const char ∗ *aTag*, int32 & *aLogInfo*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

Allows the logging level to be queried for a particular logging tag. A larger log level will result in more messages being logged.

In the asynchronous response, this should return the log level along with an indication of where the level was inherited (i.e., the ancestor tag).

Parameters

aTag Specifies the logger tree tag where the log level should be retrieved.

aLogInfo an output parameter which will be filled in with the log level information.

aContextData Optional opaque data that will be passed back to the user with the command response

Exceptions

memory_error leaves on memory allocation error.

Returns

A unique command id for asynchronous completion

4.2.3.7 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::GetSDKInfo (PVSDKInfo & *aSDKInfo*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

Returns version information about the SDK

Parameters

aSDKInfo A reference to a PVSDKInfo structure which contains the product label and date

aContextData Optional opaque data that will be passed back to the user with the command response This method can leave with one of the following error codes PVMFErrNoMemory if the SDK failed to allocate memory during this operation

Returns

A unique command id for asynchronous completion

4.2.3.8 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::GetSDKModuleInfo (PVSDKModuleInfo & *aSDKModuleInfo*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

Returns information about all modules currently used by the SDK.

Parameters

aSDKModuleInfo A reference to a PVSDKModuleInfo structure which contains the number of modules currently used by pv2way SDK and the PV UID and description string for each module. The PV UID and description string for modules will be returned in one string buffer allocated by the client. If the string buffer is not large enough to hold the all the module's information, the information will be written up to the length of the buffer and truncated.

aContextData Optional opaque data that will be passed back to the user with the command response This method can leave with one of the following error codes PVMFErrNoMemory if the SDK failed to allocate memory during this operation

Returns

A unique command id for asynchronous completion

4.2.3.9 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::GetState (PV2WayState & *aState*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This function returns the current state of the pv2way. Application may use this info for updating display or determine if the pv2way is ready for the next request.

Parameters

aState Reflects the state of the PV 2Way engine when the command was received.

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

value indicating the current pv2way state

4.2.3.10 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::Init (PV2WayInitInfo & *aInitInfo*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This function is valid only in the EIdle state. It is a no-op when invoked in any other state. It causes the pv2way to transition to the ESetup state. The terminal remains in the EInitializing state during the transition.

While initializing, the pv2way tries to allocate system resources needed for a two-way call. If it fails for some reason, and the pv2way reverts to the EIdle state. All the resources are de-allocated.

Parameters

aInitInfo A reference to a CPV2WayInitInfo structure which contains the capabilities of the applications sinks and sources to handle compressed and uncompressed formats.

aContextData Optional opaque data that will be passed back to the user with the command response This method can leave with one of the following error codes PVMFErrArgument if more tx and rx codecs are set than engine can handle, or the mandatory codecs are not in the list. PVMFErrNotSupported if the format of the sources/sinks is incomtible with what the SDK can handle PVMFErrInvalidState if invoked in the incorrect state PVMFErrNoMemory if the SDK failed to allocate memory during this operation

Returns

A unique command id for asynchronous completion

4.2.3.11 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::Pause (PV2WayDirection *aDirection*, PVTrackId *aTrackId*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

For an incoming track this function pauses sending media to the sink (output device) and stops the sink.

For outgoing, it pauses the sending of media from the source and stops the source.

Parameters

aDirection Specifies the direction of the track - incoming or outgoing

aTrackId Specifies which track is to be paused.

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.2.3.12 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::QueryInterface (const PVUuid & *aUuid*, PVInterface ∗& *aInterfacePtr*, OsclAny ∗ *aContext* = `NULL`) `[pure virtual]`

This API is to allow for extensibility of the pv2way interface. It allows a caller to ask for an instance of a particular interface object to be returned. The mechanism is analogous to the COM IUnknown method. The interfaces are identified with an interface ID that is a UUID as in DCE and a pointer to the interface object is returned if it is supported. Otherwise the returned pointer is NULL. TBD: Define the UIID, InterfacePtr structures

Parameters

aUuid The UUID of the desired interface

aInterfacePtr The output pointer to the desired interface

Exceptions

not_supported leaves if the specified interface id is not supported.

4.2.3.13 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::RemoveDataSink (PVMFNodeInterface & *aDataSink*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This function unbinds a previously added sink.

Parameters

aDataSink pointer to the media sink node

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.2.3.14 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::RemoveDataSource (PVMFNodeInterface & *aDataSource*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This function unbinds a previously added source.

Parameters

aDataSource pointer to the media source node

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.2.3.15 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::RemoveLogAppender (const char ∗ *aTag*, OsclSharedPtr< PVLoggerAppender > & *aAppender*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

Allows a logging appender to be removed from the logger tree at the point specified by the input tag. The input tag cannot be NULL.

Parameters

aTag Specifies the logger tree tag where the appender should be removed.

aAppender The log appender to remove. Must be a reference to the same object that was set.

aContextData Optional opaque data that will be passed back to the user with the command response

Exceptions

memory_error leaves on memory allocation error.

Returns

A unique command id for asynchronous completion

4.2.3.16 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::Reset (OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This function is valid only in the ESetup and EInitializing state. It is a no-op when invoked in the EIdle state and returns PVMFErrInvalidState if invoked in any other state.

It causes the pv2way to transition back to the EIdle state. The terminal remains in the EResetting state during the transition.

While resetting, the pv2way de-allocates all resources resources that had been previously allocated. When it completes, ResetComplete is called and the pv2way reverts to the EIdle state.

Parameters

aContextData Optional opaque data that will be passed back to the user with the command response This method can leave with one of the following error codes PVMFErrInvalidState if invoked in the incorrect state PVMFErrNoMemory if the SDK failed to allocate memory during this operation

Returns

A unique command id for asynchronous completion

4.2.3.17 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::Resume (PV2WayDirection *aDirection*, PVTrackId *aTrackId*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

Resume a previously paused incoming or outgoing track. For incoming, this function starts resumes playing out the media to the appropriate sink based on the current settings. For outgoing it resumes encoding and sending media from the source.

Parameters

aDirection Specifies the direction of the track - incoming or outgoing

aTrackId Specifies which track is to be paused.

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.2.3.18 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::SetLogAppender (const char ∗ *aTag*, OsclSharedPtr< PVLoggerAppender > & *aAppender*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

Allows a logging appender to be attached at some point in the logger tag tree. The location in the tag tree is specified by the input tag string. A single appender can be attached multiple times in the tree, but it may result in duplicate copies of log messages if the appender is not attached in disjoint portions of the tree. A logging appender is responsible for actually writing the log message to its final location (e.g., memory, file, network, etc). This API can be called anytime after creation of the terminal.

Parameters

aTag Specifies the logger tree tag where the appender should be attached.

aAppender The log appender to attach.

aContextData Optional opaque data that will be passed back to the user with the command response

Exceptions

memory_error leaves on memory allocation error.

Returns

A unique command id for asynchronous completion

4.2.3.19 virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::SetLogLevel (const char ∗ *aTag*, int32 *aLevel*, bool *aSetSubtree* = `false`, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

Allows the logging level to be set for the logging node specified by the tag. A larger log level will result in more messages being logged. A message will only be logged if its level is LESS THAN or equal to the current log level. The set_subtree flag will allow an entire subtree, with the specified tag as the root, to be reset to the specified value.

Parameters

aTag Specifies the logger tree tag where the log level should be set.

aLevel Specifies the log level to set.

aSetSubtree Specifies whether the entire subtree with aTag as the root should be reset to the log level.

aContextData Optional opaque data that will be passed back to the user with the command response

Exceptions

memory_error leaves on memory allocation error.

Returns

A unique command id for asynchronous completion

The documentation for this class was generated from the following file:

- pv_2way_interface.h

4.3 CPV2WayProxyFactory Class Reference

```
#include <pv_2way_proxy_factory.h>
```

Static Public Member Functions

- static OSCL_IMPORT_REF void Init ()
- static OSCL_IMPORT_REF void Cleanup ()
- static OSCL_IMPORT_REF CPV2WayInterface ∗ CreateTerminal (TPVTerminalType aTerminalType, PVCommandStatusObserver ∗aCmdStatusObserver, PVInformationalEventObserver ∗aInfoEventObserver, PVErrorEventObserver ∗aErrorEventObserver)
- static OSCL_IMPORT_REF void DeleteTerminal (CPV2WayInterface ∗terminal)

4.3.1 Member Function Documentation

4.3.1.1 static OSCL_IMPORT_REF void CPV2WayProxyFactory::Cleanup () `[static]`

4.3.1.2 static OSCL_IMPORT_REF CPV2WayInterface∗ CPV2WayProxyFactory::CreateTerminal (TPVTerminalType *aTerminalType*, PVCommandStatusObserver ∗ *aCmdStatusObserver*, PVInformationalEventObserver ∗ *aInfoEventObserver*, PVErrorEventObserver ∗ *aErrorEventObserver*) `[static]`

Creates an instance of a terminal of a particular type. Initially, this will support 324m type terminals.

Parameters

aTerminalType the type of terminal to be created.

aCmdStatusObserver the observer for command status

aInfoEventObserver the observer for unsolicited informational events

aErrorEventObserver the observer for unsolicited error events

Returns

A pointer to a terminal or leaves if the type is invalid or the system is out of resources

4.3.1.3 static OSCL_IMPORT_REF void CPV2WayProxyFactory::DeleteTerminal (CPV2WayInterface ∗ *terminal*) `[static]`

This function allows the application to delete an instance of a terminal and reclaim all allocated resources. A terminal should be deleted only in the EIdle state. An attempt to delete a terminal in any other state will result in unpredictable behavior.

Parameters

terminal the terminal to be deleted.

4.3.1.4 static OSCL_IMPORT_REF void CPV2WayProxyFactory::Init () `[static]`

The documentation for this class was generated from the following file:

- pv_2way_proxy_factory.h

4.4 CPVLogicalChannelIndication Class Reference

```
#include <pv_2way_h324m_types.h>
```

Public Member Functions

- OSCL_IMPORT_REF CPVLogicalChannelIndication (TPVChannelId channelId)
- OSCL_IMPORT_REF ∼CPVLogicalChannelIndication ()
- OSCL_IMPORT_REF TPVChannelId GetChannelId ()
- OSCL_IMPORT_REF void addRef ()
- OSCL_IMPORT_REF void removeRef ()

4.4.1 Constructor & Destructor Documentation

4.4.1.1 OSCL_IMPORT_REF CPVLogicalChannelIndication::CPVLogicalChannelIndication (TPVChannelId *channelId*)

4.4.1.2 OSCL_IMPORT_REF CPVLogicalChannelIndication::∼CPVLogicalChannelIndication ()

4.4.2 Member Function Documentation

4.4.2.1 OSCL_IMPORT_REF void CPVLogicalChannelIndication::addRef ()

4.4.2.2 OSCL_IMPORT_REF TPVChannelId CPVLogicalChannelIndication::GetChannelId ()

4.4.2.3 OSCL_IMPORT_REF void CPVLogicalChannelIndication::removeRef ()

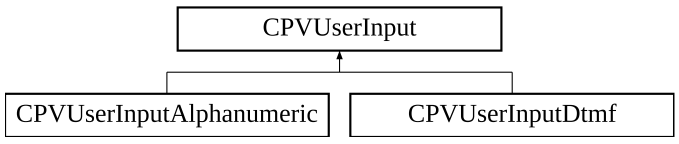
The documentation for this class was generated from the following file:

- pv_2way_h324m_types.h

4.5 CPVUserInput Class Reference

`#include <pv_2way_h324m_types.h>`

Inheritance diagram for CPVUserInput:



Public Member Functions

- OSCL_IMPORT_REF CPVUserInput ()
- virtual ∼CPVUserInput ()
- virtual PV2WayUserInputType GetType ()=0
- void addRef ()
- void removeRef ()

4.5.1 Detailed Description

CPVUserInput class Base class for User Input mesages

4.5.2 Constructor & Destructor Documentation

4.5.2.1 OSCL_IMPORT_REF CPVUserInput::CPVUserInput ()

Constructor of CPVUserInputDtmf class.

4.5.2.2 virtual CPVUserInput::∼CPVUserInput () `[inline, virtual]`

Virtual destructor

4.5.3 Member Function Documentation

4.5.3.1 void CPVUserInput::addRef () `[inline]`

4.5.3.2 virtual PV2WayUserInputType CPVUserInput::GetType () `[pure virtual]`

Virtual function to return the user input type

Implemented in CPVUserInputDtmf, and CPVUserInputAlphanumeric.

4.5.3.3 void CPVUserInput::removeRef () `[inline]`

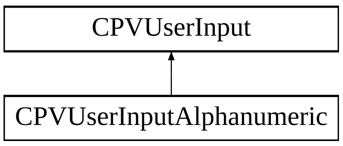
The documentation for this class was generated from the following file:

- pv_2way_h324m_types.h

4.6 CPVUserInputAlphanumeric Class Reference

```
#include <pv_2way_h324m_types.h>
```

Inheritance diagram for CPVUserInputAlphanumeric:



Public Member Functions

- OSCL_IMPORT_REF CPVUserInputAlphanumeric (uint8 ∗apInput, uint16 aLen)
- OSCL_IMPORT_REF ∼CPVUserInputAlphanumeric ()
- OSCL_IMPORT_REF PV2WayUserInputType GetType ()
- OSCL_IMPORT_REF uint8 ∗ GetInput ()
- OSCL_IMPORT_REF uint16 GetLength ()

Protected Attributes

- uint8 ∗ ipInput
- uint16 iLength

4.6.1 Detailed Description

CPVUserInputAlphanumeric Class

CPVUserInputAlphanumeric class contains an alphanumeric string from an H.245 UserInputIndication message.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 OSCL_IMPORT_REF CPVUserInputAlphanumeric::CPVUserInputAlphanumeric (uint8 ∗ *apInput*, uint16 *aLen*)

Constructor of CPVUserInputAlphanumeric class.

Parameters

apInput The input alphanumeric string (T.50 encoded).

aLen The lenght of alphanumeric string in bytes

Returns

none This method can leave with one of the following error codes OsclErrGeneral memory copy failed

4.6.2.2 OSCL_IMPORT_REF CPVUserInputAlphanumeric::∼CPVUserInputAlphanumeric ()

Destructor.

4.6.3 Member Function Documentation

4.6.3.1 OSCL_IMPORT_REF uint8∗ CPVUserInputAlphanumeric::GetInput ()

Return the user input alphanumeric user input

Returns

Returns pointer to alphanumeric user input.

4.6.3.2 OSCL_IMPORT_REF uint16 CPVUserInputAlphanumeric::GetLength ()

Return the size of alphanumeric user input

Returns

Returns size of alphanumeric user input.

4.6.3.3 OSCL_IMPORT_REF PV2WayUserInputType CPVUserInputAlphanumeric::GetType () `[virtual]`

Virtual function to return the user input type

Implements CPVUserInput.

4.6.4 Field Documentation

4.6.4.1 uint16 CPVUserInputAlphanumeric::iLength `[protected]`

4.6.4.2 uint8∗ CPVUserInputAlphanumeric::ipInput `[protected]`

The input alphanumeric string.

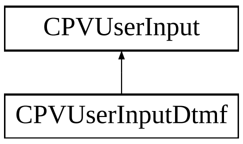
The documentation for this class was generated from the following file:

- pv_2way_h324m_types.h

4.7 CPVUserInputDtmf Class Reference

`#include <pv_2way_h324m_types.h>`

Inheritance diagram for CPVUserInputDtmf:



Public Member Functions

- OSCL_IMPORT_REF CPVUserInputDtmf (uint8 aInput, bool aUpdate, uint16 aDuration=0)
- OSCL_IMPORT_REF ∼CPVUserInputDtmf ()
- OSCL_IMPORT_REF PV2WayUserInputType GetType ()
- OSCL_IMPORT_REF uint8 GetInput ()
- OSCL_IMPORT_REF bool IsUpdate ()
- OSCL_IMPORT_REF uint16 GetDuration ()

4.7.1 Detailed Description

CPVUserInputDtmf Class

CPVUserInputDtmf class contains DTMF signal information from an H.245 UserInputIndication message.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 OSCL_IMPORT_REF CPVUserInputDtmf::CPVUserInputDtmf (uint8 *aInput*, bool *aUpdate*, uint16 *aDuration* = 0)

Constructor of CPVUserInputDtmf class.

Parameters

aInput The input DTMF tone.

aUpdate Indicates if this is an update to a continuing DTMF tone.

aDuration The duration of the update in milli-seconds. This method can leave with one of the following error codes KErrNoMemory if the SDK failed to allocate memory during this operation.

Returns

void

4.7.2.2 OSCL_IMPORT_REF CPVUserInputDtmf::∼CPVUserInputDtmf ()

Destructor.

4.7.3 Member Function Documentation

4.7.3.1 OSCL_IMPORT_REF uint16 CPVUserInputDtmf::GetDuration ()

Return the duration of the update.

Returns

Returns the duration of the update.

4.7.3.2 OSCL_IMPORT_REF uint8 CPVUserInputDtmf::GetInput ()

Return the user input DTMF tone

Returns

Returns the input DTMF tone.

4.7.3.3 OSCL_IMPORT_REF PV2WayUserInputType CPVUserInputDtmf::GetType () `[virtual]`

Virtual function to return the user input type

Implements CPVUserInput.

4.7.3.4 OSCL_IMPORT_REF bool CPVUserInputDtmf::IsUpdate ()

Return if the DTMF tone is an update

Returns

Returns if the input DTMF tone is an update.

The documentation for this class was generated from the following file:

- pv_2way_h324m_types.h

4.8 CPVVideoSpatialTemporalTradeoff Class Reference

```
#include <pv_2way_h324m_types.h>
```

Public Member Functions

- OSCL_IMPORT_REF CPVVideoSpatialTemporalTradeoff (TPVChannelId aChannelId, uint8 aTradeoff)
- OSCL_IMPORT_REF ~CPVVideoSpatialTemporalTradeoff ()
- OSCL_IMPORT_REF TPVChannelId GetChannelId ()
- OSCL_IMPORT_REF uint8 GetTradeoff ()
- OSCL_IMPORT_REF void addRef ()
- OSCL_IMPORT_REF void removeRef ()

4.8.1 Constructor & Destructor Documentation

4.8.1.1 OSCL_IMPORT_REF CPVVideoSpatialTemporalTradeoff::CPVVideoSpatialTemporalTradeoff (TPVChannelId *aChannelId*, uint8 *aTradeoff*)

4.8.1.2 OSCL_IMPORT_REF CPVVideoSpatialTemporalTradeoff::~CPVVideoSpatialTemporalTradeoff ()

4.8.2 Member Function Documentation

4.8.2.1 OSCL_IMPORT_REF void CPVVideoSpatialTemporalTradeoff::addRef ()

4.8.2.2 OSCL_IMPORT_REF TPVChannelId CPVVideoSpatialTemporalTradeoff::GetChannelId ()

4.8.2.3 OSCL_IMPORT_REF uint8 CPVVideoSpatialTemporalTradeoff::GetTradeoff ()

4.8.2.4 OSCL_IMPORT_REF void CPVVideoSpatialTemporalTradeoff::removeRef ()

The documentation for this class was generated from the following file:

- pv_2way_h324m_types.h

4.9 H324MConfigInterface Class Reference

`#include <tsc_h324m_config_interface.h>`

Public Member Functions

- virtual void SetObserver (H324MConfigObserver ∗aObserver)=0
- virtual PVMFCommandId SetMultiplexLevel (TPVH223Level aLevel, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetMaxSduSize (TPVAdaptationLayer aLayer, int32 aSize, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetMaxSduSizeR (TPVAdaptationLayer aLayer, int32 aSize, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetCodecPreference (Oscl_Vector< PVMFFormatType, OsclMemAllocator > &aIncomingAudio, Oscl_Vector< PVMFFormatType, OsclMemAllocator > &aIncomingVideo, Oscl_Vector< PVMFFormatType, OsclMemAllocator > &aOutGoingAudio, Oscl_Vector< PVMFFormatType, OsclMemAllocator > &aOutGoingVideo, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetFormatSpecificInfo (PVMFFormatType aMediaFormat, const uint8 ∗apFormatSpecificInfo, uint32 aFormatSpecificInfoLen, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetAl2SequenceNumbers (int32 aSeqNumWidth, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetAl3ControlFieldOctets (int32 aCfo, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetMaxPduSize (int32 aMaxPduSize, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetTerminalType (uint8 aTerminalType, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetALConfiguration (TPVMediaType_t aMediaType, TPVAdaptationLayer aLayer, bool aAllow, bool aUse=true, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SendRme (OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetMaxMuxPduSize (int32 aRequestMaxMuxPduSize, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetMaxMuxCcsrlSduSize (int32 aMaxCcsrlSduSize, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId FastUpdate (PVMFNodeInterface &aTrack, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SendRtd (OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetVendor (uint8 cc, uint8 ext, uint32 mc, const uint8 ∗aProduct, uint16 aProductLen, const uint8 ∗aVersion, uint16 aVersionLen, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SendEndSession (OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetEndSessionTimeout (uint32 aTimeout, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetTimerCounter (TPVH324TimerCounter aTimerCounter, uint8 aSeries, uint32 aSeriesOffset, uint32 aValue, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetVideoResolutions (TPVDirection aDirection, Oscl_Vector< PVMFVideoResolutionRange, OsclMemAllocator > &aResolutions, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SendVendorId (OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SendVideoTemporalSpatialTradeoffCommand (TPVChannelId aLogicalChannel, uint8 aTradeoff, OsclAny ∗aContextData=NULL)=0

- virtual PVMFCommandId SendVideoTemporalSpatialTradeoffIndication (TPVChannelId aLogicalChannel, uint8 aTradeoff, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SendLogicalChannelActiveIndication (TPVChannelId aLogicalChannel, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SendLogicalChannelInactiveIndication (TPVChannelId aLogicalChannel, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SendSkewIndication (TPVChannelId aLogicalChannel1, TPVChannelId aLogicalChannel2, uint16 aSkew, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetLogicalChannelBufferingMs (uint32 aInBufferingMs, uint32 aOutBufferingMs, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SendUserInput (CPVUserInput ∗user_input, OsclAny ∗aContextData=NULL)=0
- virtual PVMFCommandId SetWnsrp (const bool aEnableWnsrp, OsclAny ∗aContextData=NULL)=0

4.9.1 Member Function Documentation

4.9.1.1 virtual PVMFCommandId H324MConfigInterface::FastUpdate (PVMFNodeInterface & *aTrack*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API may be called only after the media source has been successfully added to the pv2way engine. It causes the 2way engine to immediately send out a fast update frame specific to the media type identified by the aTrack parameter.

Parameters

aContextData Optional opaque data that will be passed back to the user with the command response

aTrack The identifier for the track

Returns

A unique command id for asynchronous completion

4.9.1.2 virtual PVMFCommandId H324MConfigInterface::SendEndSession (OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

Sends an end session command to the peer. Only to be used for testing purposes.

Parameters

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.3 virtual PVMFCommandId H324MConfigInterface::SendLogicalChannelActiveIndication (TPVChannelId *aLogicalChannel*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to send a logicalChannelActive indication to the peer. It is an indication to the peer that the channel that was paused and for which it received logicalChannelInactive indication is now ready to send data on the channel defined by Logical channel Id

4.9.1.4 virtual PVMFCommandId H324MConfigInterface::SendLogicalChannelInactiveIndication (TPVChannelId *aLogicalChannel*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to send a logicalChannelInactive indication to the peer. It is an indication to the peer that the channel has been paused the channel and will not send any data on the channel defined by Logical channel Id

4.9.1.5 virtual PVMFCommandId H324MConfigInterface::SendRme (OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to specify whether Request Multiplex Entry is sent to the remote terminal after TCS

Parameters

aSendRme If true, RME is sent to the peer after TCS

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.6 virtual PVMFCommandId H324MConfigInterface::SendRtd (OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

Sends a Round Trip Determination message to the peer and indicates the round trip delay to the caller on completion of the command. The round trip delay is stored in 4 bytes in the local buffer of the completion event in network byte order.

Parameters

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.7 virtual PVMFCommandId H324MConfigInterface::SendSkewIndication (TPVChannelId *aLogicalChannel1*, TPVChannelId *aLogicalChannel2*, uint16 *aSkew*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to send a SkewIndication to the peer. Skew is measured in milliseconds, and indicates the maximum number of milliseconds that the data on logicalChannel2 is delayed from the data on logicalChannel1 as delivered to the network transport.

4.9.1.8 virtual PVMFCommandId H324MConfigInterface::SendUserInput (CPVUserInput ∗ *user_input*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

Causes the pv2way to send the specified user input to the remote terminal using control channel. The user input can be either DTMF ot Alphanumeric

Parameters

user_input A pointer to either CPVUserInputDtmf or CPVUserInputAlphanumeric

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.9 virtual PVMFCommandId H324MConfigInterface::SendVendorId (OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to send the vendor id info to the peer. Note: Calling this API during call-setup negotiations can affect the time for call-setup adversely.

4.9.1.10 virtual PVMFCommandId H324MConfigInterface::SendVideoTemporalSpatialTradeoffCommand (TPVChannelId *aLogicalChannel*, uint8 *aTradeoff*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to send a videoTemporalSpatialTradeOff command to the peer. It is a request to the remote encoder to adjust its encoding in accordance with the tradeoff value. A value of 0 indicates a high spatial resolution and a value of 31 indicates a high frame rate. The values from 0 to 31 indicate monotonically a higher frame rate. Actual values do not correspond to precise values of spatial resolution or frame rate.

4.9.1.11 virtual PVMFCommandId H324MConfigInterface::SendVideoTemporalSpatialTradeoffIndication (TPVChannelId *aLogicalChannel*, uint8 *aTradeoff*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to send a videoTemporalSpatialTradeOff command to the peer. It is an indication to the remote decoder that the local encoder has adjusted its encoding parameters according to the tradeoff value. A value of 0 indicates a high spatial resolution and a value of 31 indicates a high frame rate. The values from 0 to 31 indicate monotonically a higher frame rate. Actual values do not correspond to precise values of spatial resolution or frame rate.

4.9.1.12 virtual PVMFCommandId H324MConfigInterface::SetAl2SequenceNumbers (int32 *aSeqNumWidth*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to specify the sequence number field for AL2

Parameters

aSeqNumWidth The number of octets to use for AL2 sequence numbers. Allowed values are 0, 1.

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.13 virtual PVMFCommandId H324MConfigInterface::SetAl3ControlFieldOctets (int32 *aCfo*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to specify the control field octets field for AL3

Parameters

aCfo The number of octets to use for AL3 CFO. Allowed values are 0, 1, 2.
aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.14 virtual PVMFCommandId H324MConfigInterface::SetALConfiguration (TPVMediaType_t *aMediaType*, TPVAdaptationLayer *aLayer*, bool *aAllow*, bool *aUse* = `true`, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to specify the allowable adaptation layers for audio and video. By default AL2 is allowed for audio and AL2, AL3 are allowed for video

Parameters

aMediaType The media type, i.e audio, video, data
aLayer The adaptation layer
aAllow Is this adaptation layer allowed for this media type ?
aUse Is this adaptation layer used for video media type ?
aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.15 virtual PVMFCommandId H324MConfigInterface::SetCodecPreference (Oscl_Vector< PVMFFormatType, OsclMemAllocator > & *aIncomingAudio*, Oscl_Vector< PVMFFormatType, OsclMemAllocator > & *aIncomingVideo*, Oscl_Vector< PVMFFormatType, OsclMemAllocator > & *aOutGoingAudio*, Oscl_Vector< PVMFFormatType, OsclMemAllocator > & *aOutGoingVideo*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to specify the preference order for supported media codecs. If input vector has elements, then only these elements are used in terminal capabilities exchange.

Parameters

aIncomingAudio The incoming audio decoder preference order
aIncomingVideo The incoming video decoder preference order
aOutgoingAudio The outgoing audio encoder preference order
aOutgoingVideo The outgoing video decoder preference order
aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.16 virtual PVMFCommandId H324MConfigInterface::SetEndSessionTimeout (uint32 *aTimeout*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

Sets the disconnect timeout interval.

Parameters

aTimeout The timeout value in seconds

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.17 virtual PVMFCommandId H324MConfigInterface::SetFormatSpecificInfo (PVMFFormatType *aMediaFormat*, const uint8 ∗ *apFormatSpecificInfo*, uint32 *aFormatSpecificInfoLen*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to specify the format specific info for supported media encoders.

Parameters

aMediaFormat The outgoing media format

apFormatSpecificInfo The format specific info

aFormatSpecificInfoLen The length of the format specific info in bytes

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.18 virtual PVMFCommandId H324MConfigInterface::SetLogicalChannelBufferingMs (uint32 *aInBufferingMs*, uint32 *aOutBufferingMs*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to configure the logical channel buffer sizes for incoming and outgoing logical channels.

Parameters

aDirection The direction (Rx or Tx).

aBufferingMs The amount of buffering in milliseconds.

aContextData Optional opaque data that will be passed back to the user with the command response

4.9.1.19 virtual PVMFCommandId H324MConfigInterface::SetMaxMuxCcsrlSduSize (int32 *aMaxCcsrlSduSize*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API sets the max ccsrl sdu size

Parameters

aMaxCcsrlSduSize The max ccsrl sdu size *aContextData* Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.20 virtual PVMFCommandId H324MConfigInterface::SetMaxMuxPduSize (int32 *aRequestMaxMuxPduSize*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API causes a maxMuxPduSize request to be sent to the remote terminal if set to a valid value (64 - 255). This is done after TCS if the remote terminal supports the maxMuxPduCapability

Parameters

aRequestMaxMuxPduSize The max mux pdu size

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.21 virtual PVMFCommandId H324MConfigInterface::SetMaxPduSize (int32 *aMaxPduSize*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to limit the size of the outgoing h223 pdus

Parameters

aMaxPduSize The max pdu size

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.22 virtual PVMFCommandId H324MConfigInterface::SetMaxSduSize (TPVAdaptationLayer *aLayer*, int32 *aSize*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to specify maximum outgoing sdu sizes for each adaptation layer

Parameters

aLayer The h223 adaptation layer type

aSize The sdu size

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.23 virtual PVMFCommandId H324MConfigInterface::SetMaxSduSizeR (TPVAdaptationLayer *aLayer*, int32 *aSize*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to specify maximum incoming sdu sizes for each adaptation layer. This is indicated to the peer via the TCS

Parameters

aLayer The h223 adaptation layer type

aSize The sdu size

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.24 virtual PVMFCommandId H324MConfigInterface::SetMultiplexLevel (TPVH223Level *aLevel*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to specify the starting H223 multiplex level

Parameters

aLevel The starting H223 multiplex level. Note that the final level that is neotiated will depend on the starting level of the peer

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.25 virtual void H324MConfigInterface::SetObserver (H324MConfigObserver ∗ *aObserver*) `[pure virtual]`

This API allows the user to specify separate observers for the 324m interface. Otherwise, the default observers will be used

Parameters

aObserver the observer for command status and for unsolicited informational events

4.9.1.26 virtual PVMFCommandId H324MConfigInterface::SetTerminalType (uint8 *aTerminalType*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to specify the terminal type that is advertized to the peer. This can be used to force the local terminal to be master/slave when communicating with a peer 324m terminal for testing purposes.

Parameters

aTerminalType The terminal type

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.27 virtual PVMFCommandId H324MConfigInterface::SetTimerCounter (TPVH324TimerCounter *aTimerCounter*, uint8 *aSeries*, uint32 *aSeriesOffset*, uint32 *aValue*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

Sets an H.324 timer/counter value. This should be called before ConnectL is invoked. The supported timers are: T106 Master Slave Determination (in units of 1s) T101 Capability Exchange (in units of 1s) T103 Uni-directional and Bi-directional Logical Channel Signalling (in units of 1s) T108 Close Logical Channel (in units of 1s) T104 H.223 Multiplex Table (in units of 1s) T109 Mode Request (in units of 1s) T105 Round Trip Delay (in units of 1s) T107 Request Multiplex Entry (in units of 100ms) T401 SRP retransmission (in units of 100ms) The supported counters are: N100 H245 (TCS, MSD) N401 SRP retransmission

Parameters

aTimerCounter Identifies whether a timer or counter is being set.

aSeries Identifies the H.324 timer/counter series.

aSeriesOffset Specifies the offset within a particular series. E.g. aTimerCounter=EH324Timer, aSeries=1, aSeriesOffset=1 indicates T101. aTimerCounter=EH324Timer, aSeries=4, aSeriesOffset=1 indicates T401. aTimerCounter=EH324Counter, aSeries=4, aSeriesOffset=1 indicates T401.

aValue The new value for the H.324 timer/counter

aContextData Optional opaque data that will be passed back to the user with the command response

4.9.1.28 virtual PVMFCommandId H324MConfigInterface::SetVendor (uint8 *cc*, uint8 *ext*, uint32 *mc*, const uint8 ∗ *aProduct*, uint16 *aProductLen*, const uint8 ∗ *aVersion*, uint16 *aVersionLen*, OsclAny ∗ *aContextData* = `NULL`) `[pure virtual]`

Sets the vendor identification data. This does not cause the stack to issue a vendor identifiation request. Set to NULL to disable sending vendor id. If set to a valid parameter before Connect, it will cause the stack to automatically send it along with the TCS message.

Parameters

cc T35 Country code

ext T35 Extension

mc T35 Manufacturer code

aProduct Product number

aVersion Version number

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

4.9.1.29 virtual PVMFCommandId H324MConfigInterface::SetVideoResolutions (TPVDirection *aDirection*, Oscl_Vector< PVMFVideoResolutionRange, OsclMemAllocator > & *aResolutions*, OsclAny * *aContextData* = `NULL`) `[pure virtual]`

This API allows the user to specify the supported resolutions for video for transmit and receive.

Parameters

aDirection The direction (Tx/Rx) for which the capability is specified.

aResolutions An array of resolutions.

aContextData Optional opaque data that will be passed back to the user with the command response

4.9.1.30 virtual PVMFCommandId H324MConfigInterface::SetWnsrp (const bool *aEnableWnsrp*, OsclAny * *aContextData* = `NULL`) `[pure virtual]`

Causes the pv2way to send the specified user input to the remote terminal using control channel.

Parameters

aEnableWnsrp Boolean whether to enable Wnsrp or not

aContextData Optional opaque data that will be passed back to the user with the command response

Returns

A unique command id for asynchronous completion

The documentation for this class was generated from the following file:

- tsc_h324m_config_interface.h

4.10 H324MConfigObserver Class Reference

```
#include <tsc_h324m_config_interface.h>
```

Public Member Functions

- virtual ~H324MConfigObserver ()
- virtual void H324MConfigCommandCompletedL (PVMFCmdResp &aResponse)=0
- virtual void H324MConfigHandleInformationalEventL (PVMFAsyncEvent &aNotification)=0

4.10.1 Constructor & Destructor Documentation

4.10.1.1 virtual H324MConfigObserver::~H324MConfigObserver () `[inline, virtual]`

4.10.2 Member Function Documentation

4.10.2.1 virtual void H324MConfigObserver::H324MConfigCommandCompletedL (PVMFCmdResp & *aResponse*) `[pure virtual]`

4.10.2.2 virtual void H324MConfigObserver::H324MConfigHandleInformationalEventL (PVMFAsyncEvent & *aNotification*) `[pure virtual]`

The documentation for this class was generated from the following file:

- tsc_h324m_config_interface.h

4.11 H324MReverseParametersExtensionInterface Class Reference

`#include <tsc_h324m_config_interface.h>`

Public Member Functions

- virtual const PvmfMimeString ∗ GetFormatCapabilities ()=0
- virtual void SetPortTag (int32 aPortTag)=0
- virtual int32 GetPortTag () const =0

4.11.1 Detailed Description

Extension interface to indicate reverse logical channel parameters to the user of the stack node

4.11.2 Member Function Documentation

4.11.2.1 virtual const PvmfMimeString∗ H324MReverseParametersExtensionInterface::GetFormatCapabilities () `[pure virtual]`

Returns

The reverse media format and capabilities.

4.11.2.2 virtual int32 H324MReverseParametersExtensionInterface::GetPortTag () const `[pure virtual]`

Returns

The port tag for the reverse channel

4.11.2.3 virtual void H324MReverseParametersExtensionInterface::SetPortTag (int32 *aPortTag*) `[pure virtual]`

Parameters

aPortTag The port tag to use for the reverse channel if it is accepted.

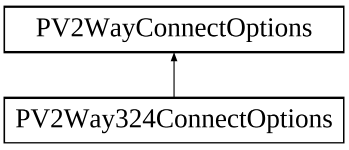
The documentation for this class was generated from the following file:

- tsc_h324m_config_interface.h

4.12 PV2Way324ConnectOptions Class Reference

`#include <pv_2way_h324m_types.h>`

Inheritance diagram for PV2Way324ConnectOptions:



Public Member Functions

- PV2Way324ConnectOptions (uint32 aDisconnectTimeoutInterval)
- PV2Way324ConnectOptions ()
- virtual ~PV2Way324ConnectOptions ()
- virtual void GetConnectInfoClassName (OSCL_wString &aClassName)

Data Fields

- uint32 iDisconnectTimeoutInterval

4.12.1 Detailed Description

PV2Way324ConnectOptions Class

PV2Way324ConnectOptions implements the PV2WayConnectOptions interface and is used for 324M specific initialization.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 PV2Way324ConnectOptions::PV2Way324ConnectOptions (uint32 *aDisconnectTimeoutInterval*) `[inline]`

Constructor

Parameters

disconnectTimeout The interval to wait after initiating a disconnect before stopping signalling

4.12.2.2 PV2Way324ConnectOptions::PV2Way324ConnectOptions () `[inline]`

4.12.2.3 virtual PV2Way324ConnectOptions::∼PV2Way324ConnectOptions () `[inline, virtual]`

4.12.3 Member Function Documentation

4.12.3.1 virtual void PV2Way324ConnectOptions::GetConnectInfoClassName (OSCL_wString & *aClassName*) `[inline, virtual]`

Retrieves the class name

Parameters

aClassName A reference to an OSCL_wString, which is to hold the subclass name, this class will assign the string "CPV2Way324ConnectInfo"

Returns

void

Implements PV2WayConnectOptions.

4.12.4 Field Documentation

4.12.4.1 uint32 PV2Way324ConnectOptions::iDisconnectTimeoutInterval

The disconnect timeout interval in units of 100ms

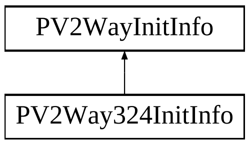
The documentation for this class was generated from the following file:

- pv_2way_h324m_types.h

4.13 PV2Way324InitInfo Class Reference

`#include <pv_2way_h324m_types.h>`

Inheritance diagram for PV2Way324InitInfo:



Public Member Functions

- virtual void GetInitInfoClassName (OSCL_wString &aClassName)
- PV2Way324InitInfo ()
- virtual ~PV2Way324InitInfo ()

Data Fields

- uint16 iMultiplexingDelayMs

4.13.1 Detailed Description

PV2Way324InitInfo Class

PV2Way324InitInfo implements the PV2Way324InitInfo interface and is used for 324M specific initialization.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 PV2Way324InitInfo::PV2Way324InitInfo () `[inline]`

4.13.2.2 virtual PV2Way324InitInfo::~PV2Way324InitInfo () `[inline, virtual]`

4.13.3 Member Function Documentation

4.13.3.1 virtual void PV2Way324InitInfo::GetInitInfoClassName (OSCL_wString & *aClassName*) `[inline, virtual]`

Retrieves the class name

Parameters

aClassName A reference to an OSCL_wString, which is to hold the subclass name, this class will assign the string "CPV2Way324InitInfo"

Returns

void

Implements PV2WayInitInfo.

4.13.4 Field Documentation

4.13.4.1 uint16 PV2Way324InitInfo::iMultiplexingDelayMs

The Multiplexing delay in Milliseconds

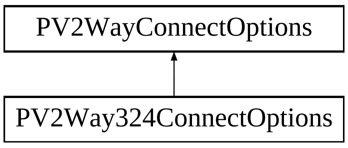
The documentation for this class was generated from the following file:

- pv_2way_h324m_types.h

4.14 PV2WayConnectOptions Class Reference

`#include <pv_2way_basic_types.h>`

Inheritance diagram for PV2WayConnectOptions:



Public Member Functions

- PV2WayConnectOptions ()
- PV2WayConnectOptions (TPVLoopbackMode aLoopbackMode, uint8 ∗aLocalId, uint32 aLocalId-Size, uint8 ∗aRemoteId, uint32 aRemoteIdSize)
- virtual ∼PV2WayConnectOptions ()
- virtual void GetConnectInfoClassName (OSCL_wString &aClassName)=0

Data Fields

- TPVLoopbackMode iLoopbackMode
- uint8 ∗ iLocalId
- uint32 iLocalIdSize
- uint8 ∗ iRemoteId
- uint32 iRemoteIdSize

4.14.1 Detailed Description

PV2WayConnectOptions Class

PV2WayConnectOptions class contains options to be specified during connect

4.14.2 Constructor & Destructor Documentation

4.14.2.1 PV2WayConnectOptions::PV2WayConnectOptions () `[inline]`

Default Constructor

4.14.2.2 PV2WayConnectOptions::PV2WayConnectOptions (TPVLoopbackMode *aLoopbackMode*, uint8 ∗ *aLocalId*, uint32 *aLocalIdSize*, uint8 ∗ *aRemoteId*, uint32 *aRemoteIdSize*) `[inline]`

Constructor

Parameters

aLoopbackMode The loopback mode to used during Connect

aLocalId,aLocalIdSize A unique octet string identifying the local terminal

aRemoteId,aRemoteIdSize A unique octet string identifying the peer (Used only in 2-Stage dialling)

Returns

void

4.14.2.3 virtual PV2WayConnectOptions::~PV2WayConnectOptions () `[inline, virtual]`

4.14.3 Member Function Documentation

4.14.3.1 virtual void PV2WayConnectOptions::GetConnectInfoClassName (OSCL_wString & *aClassName*) `[pure virtual]`

Pure virtual method that must be overridden. Retrieves class name

Parameters

aClassName A reference to an OSCL_wString, which is to hold the subclass name

Returns

void

Implemented in PV2Way324ConnectOptions.

4.14.4 Field Documentation

4.14.4.1 uint8* PV2WayConnectOptions::iLocalId

The id of the local terminal

4.14.4.2 uint32 PV2WayConnectOptions::iLocalIdSize

The size of the local id

4.14.4.3 TPVLoopbackMode PV2WayConnectOptions::iLoopbackMode

The loopback mode

4.14.4.4 uint8* PV2WayConnectOptions::iRemoteId

The id of the peer

4.14.4.5 uint32 PV2WayConnectOptions::iRemoteIdSize

The size of the remote id

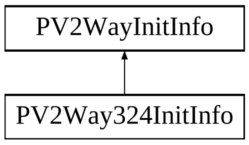
The documentation for this class was generated from the following file:

- pv_2way_basic_types.h

4.15 PV2WayInitInfo Class Reference

```
#include <pv_2way_basic_types.h>
```

Inheritance diagram for PV2WayInitInfo:



Public Member Functions

- virtual void GetInitInfoClassName (OSCL_wString &aClassName)=0
- virtual ~PV2WayInitInfo ()

Data Fields

- Oscl_Vector< PVMFFormatType, OsclMemAllocator > iOutgoingAudioFormats
- Oscl_Vector< PVMFFormatType, OsclMemAllocator > iOutgoingVideoFormats
- Oscl_Vector< PVMFFormatType, OsclMemAllocator > iIncomingAudioFormats
- Oscl_Vector< PVMFFormatType, OsclMemAllocator > iIncomingVideoFormats

4.15.1 Detailed Description

PV2WayInitInfo Class

PV2WayInitInfo is an interface required for protocols specific classes pass to the PV2WayInterface's InitL() method

4.15.2 Constructor & Destructor Documentation

4.15.2.1 virtual PV2WayInitInfo::~PV2WayInitInfo () `[inline, virtual]`

4.15.3 Member Function Documentation

4.15.3.1 virtual void PV2WayInitInfo::GetInitInfoClassName (OSCL_wString & *aClassName*) `[pure virtual]`

pure virtual method that must be overridden to return the classname of the actual subclass

Implemented in PV2Way324InitInfo.

4.15.4 Field Documentation

4.15.4.1 Oscl_Vector<PVMFFormatType, OsclMemAllocator> PV2WayInitInfo::iIncomingAudioFormats

The list of audio formats that can be received

4.15.4.2 Oscl_Vector<PVMFFormatType, OsclMemAllocator> PV2WayInitInfo::iIncomingVideoFormats

The list of video formats that can be received

4.15.4.3 Oscl_Vector<PVMFFormatType, OsclMemAllocator> PV2WayInitInfo::iOutgoingAudioFormats

The list of audio formats that can be transmitted

4.15.4.4 Oscl_Vector<PVMFFormatType, OsclMemAllocator> PV2WayInitInfo::iOutgoingVideoFormats

The list of video formats that can be transmitted

The documentation for this class was generated from the following file:

- pv_2way_basic_types.h

4.16 PV2WayTestExtensionInterface Class Reference

```
#include <pv_2way_test_extension_interface.h>
```

Public Member Functions

- virtual void addRef ()=0
- virtual void removeRef ()=0
- virtual bool queryInterface (const PVUuid &uuid, PVInterface ∗&iface)=0
- virtual bool NegotiatedFormatsMatch (Oscl_Vector< FormatCapabilityInfo, OsclMemAllocator > &iInAudFormatCapability, Oscl_Vector< FormatCapabilityInfo, OsclMemAllocator > &iOutAudFormatCapability, Oscl_Vector< FormatCapabilityInfo, OsclMemAllocator > &iInVidFormatCapability, Oscl_Vector< FormatCapabilityInfo, OsclMemAllocator > &iOutVidFormatCapability)=0
- virtual bool UsingExternalVideoDecBuffers ()=0
- virtual bool UsingExternalAudioDecBuffers ()=0

4.16.1 Member Function Documentation

4.16.1.1 virtual void PV2WayTestExtensionInterface::addRef () `[pure virtual]`

Increment reference counter for this interface.

4.16.1.2 virtual bool PV2WayTestExtensionInterface::NegotiatedFormatsMatch (Oscl_Vector< FormatCapabilityInfo, OsclMemAllocator > & *iInAudFormatCapability*, Oscl_Vector< FormatCapabilityInfo, OsclMemAllocator > & *iOutAudFormatCapability*, Oscl_Vector< FormatCapabilityInfo, OsclMemAllocator > & *iInVidFormatCapability*, Oscl_Vector< FormatCapabilityInfo, OsclMemAllocator > & *iOutVidFormatCapability*) `[pure virtual]`

4.16.1.3 virtual bool PV2WayTestExtensionInterface::queryInterface (const PVUuid & *uuid*, PVInterface ∗& *iface*) `[pure virtual]`

Query for a pointer to an instance of the interface specified by the UUID.

Parameters

uuid UUID of the interface to be queried.

iface Output parameter where a pointer to an instance of the requested interface is stored if the interface is supported.

Returns

True if successful, else false.

4.16.1.4 virtual void PV2WayTestExtensionInterface::removeRef () `[pure virtual]`

Decrement reference counter for this interface.

4.16.1.5 virtual bool PV2WayTestExtensionInterface::UsingExternalAudioDecBuffers () `[pure virtual]`

4.16.1.6 virtual bool PV2WayTestExtensionInterface::UsingExternalVideoDecBuffers () `[pure virtual]`

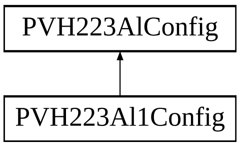
The documentation for this class was generated from the following file:

- pv_2way_test_extension_interface.h

4.17 PVH223Al1Config Class Reference

```
#include <pv_2way_h324m_types.h>
```

Inheritance diagram for PVH223Al1Config:



Public Member Functions

- PVH223AlIndex IsA () const

Data Fields

- bool iFramed

4.17.1 Detailed Description

PVH223Al1Config class

This class defines configuration information for H.223 Adaptation Layer 1

4.17.2 Member Function Documentation

4.17.2.1 PVH223AlIndex PVH223Al1Config::IsA () const `[inline, virtual]`

Implements PVH223AlConfig.

References PVH223AlConfig::PVH223_AL1.

4.17.3 Field Documentation

4.17.3.1 bool PVH223Al1Config::iFramed

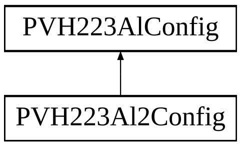
The documentation for this class was generated from the following file:

- pv_2way_h324m_types.h

4.18 PVH223Al2Config Class Reference

`#include <pv_2way_h324m_types.h>`

Inheritance diagram for PVH223Al2Config:



Public Member Functions

- PVH223AlIndex IsA () const

Data Fields

- bool iUseSequenceNumbers

4.18.1 Detailed Description

PVH223Al2Config class

This class defines configuration information for H.223 Adaptation Layer 2

4.18.2 Member Function Documentation

4.18.2.1 PVH223AlIndex PVH223Al2Config::IsA () const `[inline, virtual]`

Implements PVH223AlConfig.

References PVH223AlConfig::PVH223_AL2.

4.18.3 Field Documentation

4.18.3.1 bool PVH223Al2Config::iUseSequenceNumbers

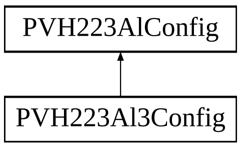
The documentation for this class was generated from the following file:

- pv_2way_h324m_types.h

4.19 PVH223Al3Config Class Reference

```
#include <pv_2way_h324m_types.h>
```

Inheritance diagram for PVH223Al3Config:



Public Member Functions

- PVH223AlIndex IsA () const

Data Fields

- uint32 iControlFieldOctets
- uint32 iSendBufferSize

4.19.1 Detailed Description

PVH223Al3Config class

This class defines configuration information for H.223 Adaptation Layer 3

4.19.2 Member Function Documentation

4.19.2.1 PVH223AlIndex PVH223Al3Config::IsA () const `[inline, virtual]`

Implements PVH223AlConfig.

References PVH223AlConfig::PVH223_AL3.

4.19.3 Field Documentation

4.19.3.1 uint32 PVH223Al3Config::iControlFieldOctets

4.19.3.2 uint32 PVH223Al3Config::iSendBufferSize

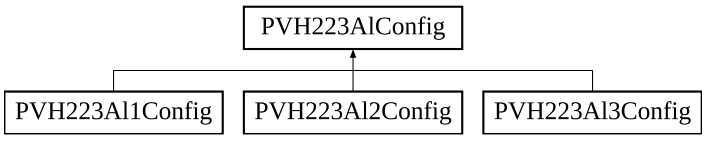
The documentation for this class was generated from the following file:

- pv_2way_h324m_types.h

4.20 PVH223AlConfig Class Reference

```
#include <pv_2way_h324m_types.h>
```

Inheritance diagram for PVH223AlConfig:



Public Types

- enum PVH223AlIndex { PVH223_AL1 = 1, PVH223_AL2 = 2, PVH223_AL3 = 4 }

Public Member Functions

- virtual ~PVH223AlConfig ()
- virtual PVH223AlIndex IsA () const =0

4.20.1 Detailed Description

PVH223AlConfig class

This is the base class for H.223 Adaptation Layer configuration

4.20.2 Member Enumeration Documentation

4.20.2.1 enum PVH223AlConfig::PVH223AlIndex

Enumerator:

PVH223_AL1

PVH223_AL2

PVH223_AL3

4.20.3 Constructor & Destructor Documentation

4.20.3.1 virtual PVH223AlConfig::~PVH223AlConfig () `[inline, virtual]`

4.20.4 Member Function Documentation

4.20.4.1 virtual PVH223AlIndex PVH223AlConfig::IsA () const `[pure virtual]`

Implemented in PVH223Al1Config, PVH223Al2Config, and PVH223Al3Config.

The documentation for this class was generated from the following file:

- pv_2way_h324m_types.h

4.21 PVMFComponentInterface Class Reference

`#include <tsc_h324m_config_interface.h>`

Public Member Functions

- PVMFComponentInterface ()
- void addRef ()
- void removeRef ()

Protected Attributes

- int32 iReferenceCount

4.21.1 Constructor & Destructor Documentation

4.21.1.1 PVMFComponentInterface::PVMFComponentInterface () `[inline]`

References iReferenceCount.

4.21.2 Member Function Documentation

4.21.2.1 void PVMFComponentInterface::addRef () `[inline]`

References iReferenceCount.

4.21.2.2 void PVMFComponentInterface::removeRef () `[inline]`

References iReferenceCount.

4.21.3 Field Documentation

4.21.3.1 int32 PVMFComponentInterface::iReferenceCount `[protected]`

Referenced by addRef(), PVMFComponentInterface(), and removeRef().

The documentation for this class was generated from the following file:

- tsc_h324m_config_interface.h

Chapter 5

File Documentation

5.1 pv_2way_basic_types.h File Reference

```
#include "pvmf_format_type.h"
#include "oscl_vector.h"
#include "oscl_mem.h"
```

Data Structures

- class PV2WayInitInfo
- class PV2WayConnectOptions

Typedefs

- typedef enum TPVTerminalType PV2WayTerminalType
- typedef enum TPVLoopbackMode PV2WayLoopbackMode
- typedef enum TPVDirection PV2WayDirection
- typedef enum TPVMediaType_t PV2WayMediaType
- typedef unsigned int PVTrackId

Enumerations

- enum TPVTerminalType { PV_323, PV_324M, PV_SIP, PV_TERMINAL_TYPE_NONE }
- enum TPVLoopbackMode { PV_LOOPBACK_NONE, PV_LOOPBACK_COMM, PV_-LOOPBACK_ENGINE, PV_LOOPBACK_MUX }
- enum TPVDirection { PV_DIRECTION_NONE = 0, INCOMING = 1, OUTGOING = 2, PV_-DIRECTION_BOTH = 3 }
- enum TPVMediaType_t {

  PV_MEDIA_NONE = 0, PV_CONTROL = 1, PV_AUDIO = 2, PV_VIDEO = 4,

  PV_DATA = 8, PV_USER_INPUT = 16, PV_MULTIPLEXED = 32, PV_MEDIA_ALL = 0xFFFF
  }

- enum PV2WayState {

  EIdle = 0, EInitializing, ESetup, EConnecting,

  EConnected, EDisconnecting, EResetting }
- enum TPVTIndicationType {

  PVT_INDICATION_INCOMING_TRACK, PVT_INDICATION_OUTGOING_TRACK, PVT_-INDICATION_DISCONNECT, PVT_INDICATION_CLOSING_TRACK,

  PVT_INDICATION_CLOSE_TRACK, PVT_INDICATION_PAUSE_TRACK, PVT_-INDICATION_RESUME_TRACK, PVT_INDICATION_INTERNAL_ERROR }

Variables

- const int PV2WayErrorStatusStart = (-10500)
- const int PV2WayDispatchError = PV2WayErrorStatusStart - 1
- const int PV2WayDisconnectError = PV2WayErrorStatusStart - 2
- const int PV2WayErrorRejected = PV2WayErrorStatusStart - 5
- const int PV2WayErrReplaced = PV2WayErrorStatusStart - 6

5.1.1 Typedef Documentation

5.1.1.1 typedef enum TPVDirection PV2WayDirection

TPVDirection Enum

TPVDirection emumerates the direction of the track.

5.1.1.2 typedef enum TPVLoopbackMode PV2WayLoopbackMode

TPVLoopbackMode Enum

TPVLoopbackMode emumerates the possible loopback options that can be used with the pv2way SDK

5.1.1.3 typedef enum TPVMediaType_t PV2WayMediaType

Enumeration of high level media types supported by the SDK

5.1.1.4 typedef enum TPVTerminalType PV2WayTerminalType

TPVTerminalType enum TPVTerminalType enumerates the possible 2-way protocols

5.1.1.5 typedef unsigned int PVTrackId

PVTrackId uniquely identifies a track for transferring audio/video in a particular direction - receive or transmit.

5.1.2 Enumeration Type Documentation

5.1.2.1 enum PV2WayState

TPV2WayState Class

An enumeration of the major states of the pv2way engine.

Enumerator:

EIdle The state immediately after the pv2way instance has been successfully created or instantiated. No resources have been allocated yet.

EInitializing The pv2way is in this state when it is initializing from the EIdle to the ESetup state. The terminal queries the available device capabilities (encode, decode, mux), acquires resources to make a two-way call (codecs, formats, memory etc) and transitions to the ESetup state when it will be ready to accept setup parameters and Connect. If initializing fails, the pv2way relinquishes the resources and reverts to the EIdle state.

ESetup The state where the pv2way instance is in the process of receiving setup parameters from the application, for encoding, multiplexing, capturing and rendering. Each time a new set of parameters is passed in, validation will take place and a status will be returned accordingly. A valid data source and data sink for the communications port are to be added to the terminal in this state before it can be transitioned to the Econnecting state. Media sources and sinks can also be added at this time.

EConnecting The state where the pv2way instance has received a call to start connecting. It starts communication with the remote terminal to exchange media capabilities and channel configuration in preparation for the establishment of media channels.

EConnected The state after all control signaling is completed. The terminal is now able to open media tracks for audio and video.

EDisconnecting The state where the terminal is shutting down all tracks and the multiplex.

EResetting The state where the terminal is releasing all resources and transitioning to the EIdle state.

5.1.2.2 enum TPVDirection

TPVDirection Enum

TPVDirection emumerates the direction of the track.

Enumerator:

PV_DIRECTION_NONE

INCOMING

OUTGOING

PV_DIRECTION_BOTH

5.1.2.3 enum TPVLoopbackMode

TPVLoopbackMode Enum

TPVLoopbackMode emumerates the possible loopback options that can be used with the pv2way SDK

Enumerator:

PV_LOOPBACK_NONE

PV_LOOPBACK_COMM
PV_LOOPBACK_ENGINE
PV_LOOPBACK_MUX

5.1.2.4 enum TPVMediaType_t

Enumeration of high level media types supported by the SDK

Enumerator:

PV_MEDIA_NONE
PV_CONTROL
PV_AUDIO
PV_VIDEO
PV_DATA
PV_USER_INPUT
PV_MULTIPLEXED
PV_MEDIA_ALL

5.1.2.5 enum TPVTerminalType

TPVTerminalType enum TPVTerminalType enumerates the possible 2-way protocols

Enumerator:

PV_323
PV_324M
PV_SIP
PV_TERMINAL_TYPE_NONE

5.1.2.6 enum TPVTIndicationType

TPVTIndicationType enum

Enumeration of unsolicited indications from pv2way.

Enumerator:

PVT_INDICATION_INCOMING_TRACK Indicates that the peer terminal has established an incoming track. The local buffer specifies the media type associated with the track. The first octet of the local buffer indicates the media type. The second,third and fourth octets are reserved. The four octets from five to eight are to be interpreted as a unique track id. The format type and additional capabilities are indicated using the PV2WayTrackInfoInterface extension interface.

PVT_INDICATION_OUTGOING_TRACK Indicates that the local terminal has established an outgoing track that is acceptable to the peer. The local buffer specifies the media type associated with the track. The first octet of the local buffer indicates the media type. The second,third and fourth octets are reserved. The four octets from five to eight are to be interpreted as a unique track id. The format type and additional capabilities are indicated using the PV2WayTrackInfoInterface extension interface.

PVT_INDICATION_DISCONNECT Indicates that 2way engine has ended the current telephony session. The app can now either reset the engine or make a subsequent call.

PVT_INDICATION_CLOSING_TRACK Indicates the start of unsolicited closure of an incoming/outgoing track. The PVT_INDICATION_CLOSE_TRACK indication will be sent when the track is completely close. The first octet of the local buffer indicates the direction of the track. The second and third octets indicates the track id.

PVT_INDICATION_CLOSE_TRACK Indicates an unsolicited closure of an incoming/outgoing track. Any media sink/source associated with this will be stopped and returned to the application. The first octet of the local buffer indicates the media type of the track. The second octet indicates the direction. The third octet indicates whether there is a replacement for this track available. If true, the application may add data source/sink for this track again.

PVT_INDICATION_PAUSE_TRACK Indicates that local terminal has paused an incoming track. Any media sink associated with this will be stopped. The first octet of the local buffer indicates the direction of the track. The second and third octets indicates the track id.

PVT_INDICATION_RESUME_TRACK Indicates that local terminal has resumed an incoming track. Any media sink associated with this will be restarted. The first octet of the local buffer indicates the direction of the track. The second and third octets indicates the track id.

PVT_INDICATION_INTERNAL_ERROR Indicates an internal error in the pv2way engine. The derived class provides further information about the actual error.

5.1.3 Variable Documentation

5.1.3.1 const int PV2WayDisconnectError = PV2WayErrorStatusStart - 2

The peer disconnected without endsession command

5.1.3.2 const int PV2WayDispatchError = PV2WayErrorStatusStart - 1

There was an error dispatching muxed data to the downstream node

5.1.3.3 const int PV2WayErrorRejected = PV2WayErrorStatusStart - 5

The request was rejected by the peer

5.1.3.4 const int PV2WayErrorStatusStart = (-10500)

The starting error code for 2way specific errors

5.1.3.5 const int PV2WayErrReplaced = PV2WayErrorStatusStart - 6

Signals replacement of an existing resource

5.2 pv_2way_engine_factory.h File Reference

```
#include "pv_2way_basic_types.h"
```

Data Structures

- class CPV2WayEngineFactory

5.3 pv_2way_h324m_types.h File Reference

```
#include "pv_2way_basic_types.h"
#include "pv_uuid.h"
#include "pv_interface.h"
#include "pvt_common.h"
#include "oscl_mem.h"
#include "oscl_defalloc.h"
#include "oscl_mem_mempool.h"
#include "oscl_shared_ptr.h"
#include "oscl_string_containers.h"
#include "pvmf_format_type.h"
#include "pvmf_return_codes.h"
#include "pvmf_video.h"
#include "pvmf_timestamp.h"
#include "pvmi_kvp.h"
#include "h245def.h"
```

Data Structures

- class PV2Way324InitInfo
- class PV2Way324ConnectOptions
- class PVH223AlConfig
- class PVH223Al1Config
- class PVH223Al2Config
- class PVH223Al3Config
- class CPVUserInput
- class CPVUserInputDtmf
- class CPVUserInputAlphanumeric
- class CPVLogicalChannelIndication
- class CPVVideoSpatialTemporalTradeoff

Defines

- #define PV_2WAY_MAX_USER_INPUT_FORMATS 4
- #define PV_2WAY_MAX_SKEW_MS 1000

Typedefs

- typedef enum TPVPostDisconnectOption PV2WayPostDisconnectOption
- typedef enum UserInputType PV2WayUserInputType

Enumerations

- enum TPVPostDisconnectOption { EDisconnectLine, EAnalogueTelephony }
- enum UserInputType { PV_ALPHANUMERIC = 0, PV_DTMF }

5.3.1 Define Documentation

5.3.1.1 #define PV_2WAY_MAX_SKEW_MS 1000

The maximum skew that can be taken into account for both outgoing and incoming sides

5.3.1.2 #define PV_2WAY_MAX_USER_INPUT_FORMATS 4

The maximum number of supported formats for user input

5.3.2 Typedef Documentation

5.3.2.1 typedef enum TPVPostDisconnectOption PV2WayPostDisconnectOption

TPVPostDisconnectOption Enum

TPVPostDisconnectOption emumerates the mode the peer wants to transition to after the disconnect

5.3.2.2 typedef enum UserInputType PV2WayUserInputType

UserInputType enum Enumeration of user input types

5.3.3 Enumeration Type Documentation

5.3.3.1 enum TPVPostDisconnectOption

TPVPostDisconnectOption Enum

TPVPostDisconnectOption emumerates the mode the peer wants to transition to after the disconnect

Enumerator:

EDisconnectLine

EAnalogueTelephony

5.3.3.2 enum UserInputType

UserInputType enum Enumeration of user input types

Enumerator:

PV_ALPHANUMERIC

PV_DTMF

5.4 pv_2way_interface.h File Reference

```
#include "pv_common_types.h"
#include "oscl_vector.h"
#include "pvt_common.h"
#include "pvmf_node_interface.h"
#include "pvlogger_accessories.h"
#include "pv_engine_types.h"
#include "pv_2way_basic_types.h"
#include "pv_2way_h324m_types.h"
```

Data Structures

- class CPV2WayInterface

5.5 pv_2way_proxy_factory.h File Reference

```
#include "pv_common_types.h"
#include "pv_2way_interface.h"
#include "oscl_vector.h"
#include "pvt_common.h"
#include "pvmf_node_interface.h"
#include "pvlogger_accessories.h"
#include "pv_engine_types.h"
#include "pv_2way_basic_types.h"
#include "pv_2way_h324m_types.h"
#include "pv_engine_observer.h"
```

Data Structures

- class CPV2WayProxyFactory

5.6 pv_2way_test_extension_interface.h File Reference

```
#include "oscl_base.h"
#include "oscl_string.h"
#include "oscl_refcounter_memfrag.h"
#include "pv_uuid.h"
#include "pv_interface.h"
```

Data Structures

- class PV2WayTestExtensionInterface

Defines

- #define PV2WayTestEncExtensionUUID PVUuid(0x19b53654, 0x2dd4,0x4469,0xa9,0xdb,0x86,0x28,0xa7,0x69,0x92,0

5.6.1 Define Documentation

5.6.1.1 #define PV2WayTestEncExtensionUUID PVUuid(0x19b53654, 0x2dd4,0x4469,0xa9,0xdb,0x86,0x28,0xa7,0x69,0x92,0xe3)

5.7 tsc_h324m_config_interface.h File Reference

```
#include "oscl_base.h"
#include "pvt_common.h"
#include "pv_uuid.h"
#include "pvmf_node_interface.h"
#include "pv_2way_h324m_types.h"
```

Data Structures

- class PVMFComponentInterface
- class H324MConfigObserver
- class H324MConfigInterface
- class H324MReverseParametersExtensionInterface

Defines

- #define PVH324MConfigUuid PVUuid(0x2b0b54e2,0x7079,0x46c6,0xb2,0x3e,0x04,0xff,0xd3,0x0e,0x14,0x36)
- #define PVUuidH324ComponentInterface PVUuid(0xf6b47190,0xf88d,0x4cbf,0xa6,0xf6,0xc6,0x1e,0xfe,0x98,0x05,0x

Typedefs

- typedef uint32 H324MConfigInformationalEvent
- typedef uint32 H324MConfigStatusResponse

Enumerations

- enum PVH324MIndicationType {

  PV_INDICATION_VIDEO_SPATIAL_TEMPORAL_TRADEOFF_COMMAND, PV_-INDICATION_VIDEO_SPATIAL_TEMPORAL_TRADEOFF_INDICATION, PV_-INDICATION_FAST_UPDATE, PV_INDICATION_RTD,

  PV_INDICATION_RME, PV_INDICATION_VENDOR_ID, PV_INDICATION_USER_INPUT_-CAPABILITY, PV_INDICATION_USER_INPUT,

  PV_INDICATION_SKEW, PV_INDICATION_LOGICAL_CHANNEL_ACTIVE, PV_-INDICATION_LOGICAL_CHANNEL_INACTIVE }

5.7.1 Define Documentation

5.7.1.1 #define PVH324MConfigUuid PVUuid(0x2b0b54e2,0x7079,0x46c6,0xb2,0x3e,0x04,0xff,0xd3,0x0e,0x14,0x36)

5.7.1.2 #define PVUuidH324ComponentInterface PVUuid(0xf6b47190,0xf88d,0x4cbf,0xa6,0xf6,0xc6,0x1e,0xfe,0x98,0x05,0x3f)

5.7.2 Typedef Documentation

5.7.2.1 typedef uint32 H324MConfigInformationalEvent

5.7.2.2 typedef uint32 H324MConfigStatusResponse

5.7.3 Enumeration Type Documentation

5.7.3.1 enum PVH324MIndicationType

PVH324MIndicationType enum

Enumeration of unsolicited H324m specific indications from pv2way.

Enumerator:

PV_INDICATION_VIDEO_SPATIAL_TEMPORAL_TRADEOFF_COMMAND Indicates the receipt of a videoSpatialTemporalTradeoff command from the peer. The first 2 bytes of the event local buffer indicate the logical channel (network byte order) and the 3rd byte indicates the tradeoff value.

PV_INDICATION_VIDEO_SPATIAL_TEMPORAL_TRADEOFF_INDICATION Indicates the receipt of a videoSpatialTemporalTradeoff indication from the peer. The first 2 bytes of the event local buffer indicate the logical channel (network byte order) and the 3rd byte indicates the tradeoff value.

PV_INDICATION_FAST_UPDATE Indicates a fast update message from the remote terminal. The first two bytes of the local buffer encode the logical channel number in network byte order.

PV_INDICATION_RTD Indicates an incoming RTD command.

PV_INDICATION_RME Indicates an incoming request multiplex entry command.

PV_INDICATION_VENDOR_ID Indicates an incoming vendor id indication message.

PV_INDICATION_USER_INPUT_CAPABILITY Indicates the receipt of user input capability from the remote terminal. The first byte in local buffer contains the user input formats supported by the peer. The least significant stands for basicString, 2nd bit defines support for iA5String, 3rd for generalString and 4th dtmf.

PV_INDICATION_USER_INPUT Indicates the receipt of user input from the remote terminal. Event extension interface contains pointer to CPVUserInput that holds the actual user input sequences received.

PV_INDICATION_SKEW Indicates the receipt of a an h223SkewIndication indication from the peer. The first 2 bytes of the event local buffer indicate the first logical channel, the 3rd and 4th bytes the second logical channel and the 5th and 6th bytes the value of the skew in milliseconds. All values are in network byte order.

PV_INDICATION_LOGICAL_CHANNEL_ACTIVE Indicates a Active Logical Channel message from the remote terminal. The first two bytes of the local buffer encode the logical channel number in network byte order.

PV_INDICATION_LOGICAL_CHANNEL_INACTIVE Indicates a Inactive Logical Channel message from the remote terminal. The first two bytes of the local buffer encode the logical channel number in network byte order.

Index