



PV2Way Engine

Build Version: OPENCORE_20090517

May 17, 2009

Contents

1	pv2way_engine Hierarchical Index	1
1.1	pv2way_engine Class Hierarchy	1
2	pv2way_engine Data Structure Index	2
2.1	pv2way_engine Data Structures	2
3	pv2way_engine File Index	3
3.1	pv2way_engine File List	3
4	pv2way_engine Data Structure Documentation	4
4.1	CPV2WayEngineFactory Class Reference	4
4.2	CPV2WayInterface Class Reference	6
4.3	CPV2WayProxyFactory Class Reference	15
4.4	H324MConfigInterface Class Reference	16
4.5	PV2Way324ConnectOptions Class Reference	24
4.6	PV2Way324InitInfo Class Reference	26
4.7	PV2WayConnectOptions Class Reference	28
4.8	PV2WayInitInfo Class Reference	30
4.9	PV2WayTestExtensionInterface Class Reference	32
4.10	PVH223A11Config Class Reference	33
4.11	PVH223A12Config Class Reference	34
4.12	PVH223A13Config Class Reference	35
4.13	PVH223A1Config Class Reference	36
5	pv2way_engine File Documentation	37
5.1	pv_2way_basic_types.h File Reference	37
5.2	pv_2way_engine_factory.h File Reference	42
5.3	pv_2way_h324m_interface.h File Reference	43
5.4	pv_2way_h324m_types.h File Reference	45
5.5	pv_2way_interface.h File Reference	47

5.6	pv_2way_proxy_factory.h File Reference	48
5.7	pv_2way_test_extension_interface.h File Reference	49

Chapter 1

pv2way_engine Hierarchical Index

1.1 pv2way_engine Class Hierarchy

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

CPV2WayEngineFactory	4
CPV2WayInterface	6
CPV2WayProxyFactory	15
H324MConfigInterface	16
PV2WayConnectOptions	28
PV2Way324ConnectOptions	24
PV2WayInitInfo	30
PV2Way324InitInfo	26
PV2WayTestExtensionInterface	32
PVH223AlConfig	36
PVH223Al1Config	33
PVH223Al2Config	34
PVH223Al3Config	35

Chapter 2

pv2way_engine Data Structure Index

2.1 pv2way_engine Data Structures

Here are the data structures with brief descriptions:

CPV2WayEngineFactory	4
CPV2WayInterface	6
CPV2WayProxyFactory	15
H324MConfigInterface	16
PV2Way324ConnectOptions	24
PV2Way324InitInfo	26
PV2WayConnectOptions	28
PV2WayInitInfo	30
PV2WayTestExtensionInterface	32
PVH223A11Config	33
PVH223A12Config	34
PVH223A13Config	35
PVH223A1Config	36

Chapter 3

pv2way_engine File Index

3.1 pv2way_engine File List

Here is a list of all files with brief descriptions:

pv_2way_basic_types.h	37
pv_2way_engine_factory.h	42
pv_2way_h324m_interface.h	43
pv_2way_h324m_types.h	45
pv_2way_interface.h	47
pv_2way_proxy_factory.h	48
pv_2way_test_extension_interface.h	49

Chapter 4

pv2way_engine Data Structure Documentation

4.1 CPV2WayEngineFactory Class Reference

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

Static Public Methods

- OSL_IMPORT_REF void [Init](#) ()
- OSL_IMPORT_REF void [Cleanup](#) ()
- OSL_IMPORT_REF [CPV2WayInterface](#) * [CreateTerminal](#) ([PV2WayTerminalType](#) aTerminalType, PVCommandStatusObserver *aCmdStatusObserver, PVInformationalEventObserver *aInfoEventObserver, PVErrEventObserver *aErrorEventObserver)
- OSL_IMPORT_REF void [DeleteTerminal](#) ([CPV2WayInterface](#) *terminal)

4.1.1 Member Function Documentation

4.1.1.1 OSL_IMPORT_REF void CPV2WayEngineFactory::Cleanup () [static]

4.1.1.2 OSL_IMPORT_REF [CPV2WayInterface](#)* CPV2WayEngineFactory::CreateTerminal ([PV2WayTerminalType](#) aTerminalType, PVCommandStatusObserver * aCmdStatusObserver, PVInformationalEventObserver * aInfoEventObserver, PVErrEventObserver * 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 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 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 Methods

- 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 [QueryUUID](#) (const PvmfMimeString &aMimeType, Oscl_Vector< PVUuid, BasicAlloc > &aUuids, bool aExactUuidsOnly=false, 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 PVAAsyncInformationalEvent. 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 PVAAsyncInformationalEvent. 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
 @leave This method can leave with one of the following error codes PVMFErrNotSupported if the format of the sources/sinks is incompatible 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 (OslAny * *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, OslAny * *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 (OslAny * *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
 @leave 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
 @leave 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
 @leave 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 incompatible 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 UUID, 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::QueryUUID (const
PvmfMimeString & aMimeType, Oscl_Vector< PVUuid, BasicAlloc > & aUuids, bool
aExactUuidsOnly = false, OsclAny * aContextData = NULL) [pure virtual]`

This API is to allow for extensibility of the pv2way interface. It allows a caller to ask for all UUIDs associated with a particular MIME type. If interfaces of the requested MIME type are found within the system, they are added to the UUIDs array.

Also added to the UUIDs array will be all interfaces which have the requested MIME type as a base MIME type. This functionality can be turned off.

Parameters:

- aMimeType* The MIME type of the desired interfaces
- aUuids* An array to hold the discovered UUIDs
- aExactUuidsOnly* Turns on/off the retrieval of UUIDs with aMimeType as a base type
- aContextData* Optional opaque data that will be passed back to the user with the command response

4.2.3.14 `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

4.2.3.15 `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

4.2.3.16 `virtual OSCL_IMPORT_REF PVCommandId CPV2WayInterface::RemoveLog-Appender (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

4.2.3.17 `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
 @leave 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

4.2.3.18 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

4.2.3.19 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

4.2.3.20 `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 Methods

- OSCL_IMPORT_REF void [Init](#) ()
- OSCL_IMPORT_REF void [Cleanup](#) ()
- OSCL_IMPORT_REF [CPV2WayInterface](#) * [CreateTerminal](#) ([TPVTerminalType](#) aTerminalType, PVCommandStatusObserver *aCmdStatusObserver, PVInformationalEventObserver *aInfoEventObserver, PVErrrorEventObserver *aErrorEventObserver)
- OSCL_IMPORT_REF void [DeleteTerminal](#) ([CPV2WayInterface](#) *terminal)

4.3.1 Member Function Documentation

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

4.3.1.2 OSCL_IMPORT_REF [CPV2WayInterface](#)* CPV2WayProxyFactory::CreateTerminal ([TPVTerminalType](#) aTerminalType, PVCommandStatusObserver * aCmdStatusObserver, PVInformationalEventObserver * aInfoEventObserver, PVErrrorEventObserver * 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 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 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 H324MConfigInterface Class Reference

```
#include <pv_2way_h324m_interface.h>
```

Public Methods

- virtual void [SetObservers](#) (PVCommandStatusObserver *aCmdStatusObserver, PVInformational-EventObserver *aInfoEventObserver, PVErrorEventObserver *aErrorEventObserver)=0
- virtual PVCommandId [SetMultiplexLevel](#) (PVH223Level aLevel, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SetMaxSduSize](#) (PVH223AdaptationLayer aLayer, int32 aSize, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SetMaxSduSizeR](#) (PVH223AdaptationLayer aLayer, int32 aSize, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SetOutgoingChannelConfiguration](#) (int32 aMediaTypes, [PVH223AIConfig](#) *aConfig, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SetIncomingChannelConfiguration](#) (uint32 iAudioAdaptationLayers, uint32 iVideoAdaptationLayers, uint32 iDataAdaptationLayers, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SetMaxPduSize](#) (int32 aMaxPduSize, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SetTerminalType](#) (uint8 aTerminalType, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SendRme](#) (OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SetMaxMuxPduSize](#) (int32 aRequestMaxMuxPduSize, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SetMaxMuxCcslSduSize](#) (int32 aMaxCcslSduSize, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [FastUpdate](#) (PVChannelId aChannelId, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SendRtd](#) (OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SetVendor](#) (uint8 cc, uint8 ext, uint32 mc, const uint8 *aProduct, uint16 aProductLen, const uint8 *aVersion, uint16 aVersionLen, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SendEndSession](#) (OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SetEndSessionTimeout](#) (uint32 aTimeout, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SetTimerCounter](#) (PVH324TimerCounter aTimerCounter, uint8 aSeries, uint32 aSeriesOffset, uint32 aValue, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SetVideoResolutions](#) (PVDirection aDirection, Oscl_Vector< PVMFVideo-ResolutionRange, OsclMemAllocator > &aResolutions, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SendVendorId](#) (OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SendVideoTemporalSpatialTradeoffCommand](#) (PVChannelId aLogicalChannel, uint8 aTradeoff, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SendVideoTemporalSpatialTradeoffIndication](#) (PVChannelId aLogicalChannel, uint8 aTradeoff, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SendSkewIndication](#) (PVChannelId aLogicalChannel1, PVChannelId aLogicalChannel2, uint16 aSkew, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SetLogicalChannelBufferingMs](#) (uint32 aInBufferingMs, uint32 aOutBufferingMs, OsclAny *aContextData=NULL)=0
- virtual PVCommandId [SendUserInput](#) (CPVUserInput *user_input, OsclAny *aContextData=NULL)=0

4.4.1 Detailed Description

H324MConfigInterface Class

H324MConfigInterface provides H.324m specific configuration APIs.

4.4.2 Member Function Documentation

4.4.2.1 **virtual PVCommandId H324MConfigInterface::FastUpdate (PVChannelId *aChannelId*, OsciAny * *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:

aChannelId The identifier for the track

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

Returns:

A unique command id for asynchronous completion

4.4.2.2 **virtual PVCommandId H324MConfigInterface::SendEndSession (OsciAny * *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

4.4.2.3 **virtual PVCommandId H324MConfigInterface::SendRme (OsciAny * *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

4.4.2.4 virtual PVCommandId 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

4.4.2.5 virtual PVCommandId H324MConfigInterface::SendSkewIndication (PVChannelId *aLogicalChannel1*, PVChannelId *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.

4.4.2.6 virtual PVCommandId 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 or 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

4.4.2.7 virtual PVCommandId 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.

4.4.2.8 virtual PVCommandId H324MConfigInterface::SendVideoTemporalSpatialTradeoff-Command (PVChannelId *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.

4.4.2.9 virtual PVCommandId H324MConfigInterface::SendVideoTemporalSpatialTradeoff-Indication (PVChannelId *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.

4.4.2.10 virtual PVCommandId 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

4.4.2.11 virtual PVCommandId H324MConfigInterface::SetIncomingChannelConfiguration (uint32 *iAudioAdaptationLayers*, uint32 *iVideoAdaptationLayers*, uint32 *iDataAdaptationLayers*, OsclAny * *aContextData* = NULL) [pure virtual]

This API allows the user to specify the configuration for incoming channels

Parameters:

iAudioAdaptationLayers The allowable adaptation layers for incoming audio channels specified as a bitarray

iVideoAdaptationLayers The allowable adaptation layers for incoming video channels specified as a bitarray

iDataAdaptationLayers The allowable adaptation layers for incoming data channels specified as a bitarray

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

Returns:

A unique command id for asynchronous completion

4.4.2.12 virtual PVCommandId 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

4.4.2.13 virtual PVCommandId 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

4.4.2.14 virtual PVCommandId 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

4.4.2.15 virtual PVCommandId 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

4.4.2.16 virtual PVCommandId H324MConfigInterface::SetMaxSduSize (PVH223AdaptationLayer *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

4.4.2.17 `virtual PVCommandId H324MConfigInterface::SetMaxSduSizeR (PVH223AdaptationLayer 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

4.4.2.18 `virtual PVCommandId H324MConfigInterface::SetMultiplexLevel (PVH223Level 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 negotiated 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

4.4.2.19 `virtual void H324MConfigInterface::SetObservers (PVCommandStatusObserver * aCmdStatusObserver, PVInformationalEventObserver * aInfoEventObserver, PVErrrorEventObserver * aErrorEventObserver) [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

4.4.2.20 `virtual PVCommandId H324MConfigInterface::SetOutgoingChannelConfiguration (int32 aMediaTypes, PVH223AlConfig * aConfig, OsclAny * aContextData = NULL) [pure virtual]`

This API allows the user to specify the configuration for outgoing AL channels

Parameters:

aMediaTypes Media types for which configuration is being specified

aConfig Adaptation Layer configuration

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

Returns:

A unique command id for asynchronous completion

4.4.2.21 `virtual PVCommandId H324MConfigInterface::SetTerminalType (uint8 aTerminalType, OsciAny * 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

4.4.2.22 `virtual PVCommandId H324MConfigInterface::SetTimerCounter (PVH324TimerCounter aTimerCounter, uint8 aSeries, uint32 aSeriesOffset, uint32 aValue, OsciAny * 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

4.4.2.23 `virtual PVCommandId H324MConfigInterface::SetVendor (uint8 cc, uint8 ext, uint32 mc, const uint8 * aProduct, uint16 aProductLen, const uint8 * aVersion, uint16 aVersionLen, OsciAny * aContextData = NULL) [pure virtual]`

Sets the vendor identification data. This does not cause the stack to issue a vendor identification 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

4.4.2.24 `virtual PVCommandId H324MConfigInterface::SetVideoResolutions (PVDirection
aDirection, Osci_Vector< PVMFVideoResolutionRange, OsciMemAllocator > &
aResolutions, OsciAny * 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

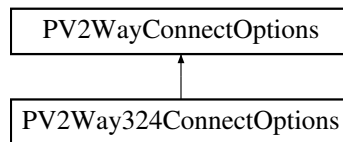
The documentation for this class was generated from the following file:

- [pv_2way_h324m_interface.h](#)

4.5 PV2Way324ConnectOptions Class Reference

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

Inheritance diagram for PV2Way324ConnectOptions::



Public Methods

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

Data Fields

- uint32 [iDisconnectTimeoutInterval](#)

4.5.1 Detailed Description

PV2Way324ConnectOptions Class

PV2Way324ConnectOptions implements the [PV2WayConnectOptions](#) interface and is used for 324M specific initialization.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 PV2Way324ConnectOptions::PV2Way324ConnectOptions (uint32 aDisconnectTimeoutInterval) [inline]

Constructor

Parameters:

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

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

4.5.2.3 `virtual PV2Way324ConnectOptions::~~PV2Way324ConnectOptions ()` [inline, virtual]

4.5.3 Member Function Documentation

4.5.3.1 `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](#).

4.5.4 Field Documentation

4.5.4.1 `uint32 PV2Way324ConnectOptions::iDisconnectTimeoutInterval`

The disconnect timeout interval in units of 100ms

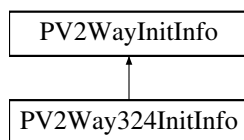
The documentation for this class was generated from the following file:

- [pv_2way_h324m_types.h](#)

4.6 PV2Way324InitInfo Class Reference

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

Inheritance diagram for PV2Way324InitInfo::



Public Methods

- virtual void [GetInitInfoClassName](#) (OSCL_wString &aClassName)
- [PV2Way324InitInfo](#) ()
- virtual [~PV2Way324InitInfo](#) ()

Data Fields

- uint16 [iMultiplexingDelayMs](#)

4.6.1 Detailed Description

PV2Way324InitInfo Class

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

4.6.2 Constructor & Destructor Documentation

4.6.2.1 [PV2Way324InitInfo::PV2Way324InitInfo](#) () [inline]

4.6.2.2 [virtual PV2Way324InitInfo::~~PV2Way324InitInfo](#) () [inline, virtual]

4.6.3 Member Function Documentation

4.6.3.1 [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](#).

4.6.4 Field Documentation

4.6.4.1 uint16 PV2Way324InitInfo::iMultiplexingDelayMs

The Multiplexing delay in Milliseconds

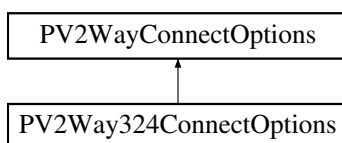
The documentation for this class was generated from the following file:

- [pv_2way_h324m_types.h](#)

4.7 PV2WayConnectOptions Class Reference

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

Inheritance diagram for PV2WayConnectOptions::



Public Methods

- [PV2WayConnectOptions \(\)](#)
- [PV2WayConnectOptions \(TPVLoopbackMode aLoopbackMode, uint8 *aLocalId, uint32 aLocalIdSize, uint8 *aRemoteId, uint32 aRemoteIdSize\)](#)
- virtual void [GetConnectInfoClassName](#) (OSCL_wString &aClassName)=0

Data Fields

- [TPVLoopbackMode iLoopbackMode](#)
- [uint8 * iLocalId](#)
- [uint32 iLocalIdSize](#)
- [uint8 * iRemoteId](#)
- [uint32 iRemoteIdSize](#)

4.7.1 Detailed Description

PV2WayConnectOptions Class

PV2WayConnectOptions class contains options to be specified during connect

4.7.2 Constructor & Destructor Documentation

4.7.2.1 PV2WayConnectOptions::PV2WayConnectOptions () [inline]

Default Constructor

4.7.2.2 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

4.7.3 Member Function Documentation

4.7.3.1 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](#).

4.7.4 Field Documentation

4.7.4.1 uint8* PV2WayConnectOptions::iLocalId

The id of the local terminal

4.7.4.2 uint32 PV2WayConnectOptions::iLocalIdSize

The size of the local id

4.7.4.3 [TPVLoopbackMode](#) PV2WayConnectOptions::iLoopbackMode

The loopback mode

4.7.4.4 uint8* PV2WayConnectOptions::iRemoteId

The id of the peer

4.7.4.5 uint32 PV2WayConnectOptions::iRemoteIdSize

The size of the remote id

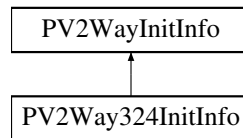
The documentation for this class was generated from the following file:

- [pv_2way_basic_types.h](#)

4.8 PV2WayInitInfo Class Reference

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

Inheritance diagram for PV2WayInitInfo::



Public Methods

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

Data Fields

- Osci_Vector< CodecSpecifier *, OsciMemAllocator > [iOutgoingAudioFormats](#)
- Osci_Vector< CodecSpecifier *, OsciMemAllocator > [iOutgoingVideoFormats](#)
- Osci_Vector< CodecSpecifier *, OsciMemAllocator > [iIncomingAudioFormats](#)
- Osci_Vector< CodecSpecifier *, OsciMemAllocator > [iIncomingVideoFormats](#)

4.8.1 Detailed Description

PV2WayInitInfo Class

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

4.8.2 Constructor & Destructor Documentation

4.8.2.1 virtual PV2WayInitInfo::~~PV2WayInitInfo () [inline, virtual]

4.8.3 Member Function Documentation

4.8.3.1 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](#).

4.8.4 Field Documentation

4.8.4.1 Osci_Vector<CodecSpecifier*, OsciMemAllocator> PV2WayInitInfo::iIncomingAudioFormats

The list of audio formats that can be received

4.8.4.2 Osci_Vector<CodecSpecifier*, OsciMemAllocator> PV2WayInitInfo::iIncomingVideo-Formats

The list of video formats that can be received

4.8.4.3 Osci_Vector<CodecSpecifier*, OsciMemAllocator> PV2WayInitInfo::iOutgoingAudio-Formats

The list of audio formats that can be transmitted

4.8.4.4 Osci_Vector<CodecSpecifier*, OsciMemAllocator> PV2WayInitInfo::iOutgoingVideo-Formats

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](#)

4.9 PV2WayTestExtensionInterface Class Reference

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

Public Methods

- virtual void [addRef](#) ()=0
- virtual void [removeRef](#) ()=0
- virtual bool [queryInterface](#) (const PVUuid &uuid, PVInterface *&iface)=0
- virtual bool [AcceptableFormatsMatch](#) (OscVector< FormatCapabilityInfo, OscMemAllocator > &iInAudFormatCapability, OscVector< FormatCapabilityInfo, OscMemAllocator > &iOutAudFormatCapability, OscVector< FormatCapabilityInfo, OscMemAllocator > &iInVidFormatCapability, OscVector< FormatCapabilityInfo, OscMemAllocator > &iOutVidFormatCapability)=0

4.9.1 Member Function Documentation

4.9.1.1 virtual bool PV2WayTestExtensionInterface::AcceptableFormatsMatch (OscVector< FormatCapabilityInfo, OscMemAllocator > &iInAudFormatCapability, OscVector< FormatCapabilityInfo, OscMemAllocator > &iOutAudFormatCapability, OscVector< FormatCapabilityInfo, OscMemAllocator > &iInVidFormatCapability, OscVector< FormatCapabilityInfo, OscMemAllocator > &iOutVidFormatCapability) [pure virtual]

4.9.1.2 virtual void PV2WayTestExtensionInterface::addRef () [pure virtual]

Increment reference counter for this interface.

4.9.1.3 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.

4.9.1.4 virtual void PV2WayTestExtensionInterface::removeRef () [pure virtual]

Decrement reference counter for this interface.

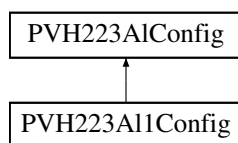
The documentation for this class was generated from the following file:

- [pv_2way_test_extension_interface.h](#)

4.10 PVH223Al1Config Class Reference

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

Inheritance diagram for PVH223Al1Config::



Public Methods

- [PVH223AlIndex IsA \(\) const](#)

Data Fields

- bool [iFramed](#)

4.10.1 Detailed Description

PVH223Al1Config class

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

4.10.2 Member Function Documentation

4.10.2.1 [PVH223AlIndex](#) PVH223Al1Config::IsA () const [inline, virtual]

Implements [PVH223AlConfig](#).

4.10.3 Field Documentation

4.10.3.1 bool PVH223Al1Config::iFramed

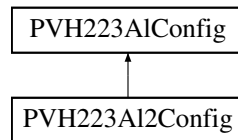
The documentation for this class was generated from the following file:

- [pv_2way_h324m_types.h](#)

4.11 PVH223Al2Config Class Reference

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

Inheritance diagram for PVH223Al2Config::



Public Methods

- [PVH223AlIndex IsA \(\) const](#)

Data Fields

- [bool iUseSequenceNumbers](#)

4.11.1 Detailed Description

PVH223Al2Config class

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

4.11.2 Member Function Documentation

4.11.2.1 [PVH223AlIndex](#) PVH223Al2Config::IsA () const [inline, virtual]

Implements [PVH223AlConfig](#).

4.11.3 Field Documentation

4.11.3.1 [bool](#) PVH223Al2Config::iUseSequenceNumbers

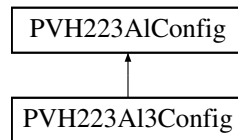
The documentation for this class was generated from the following file:

- [pv_2way_h324m_types.h](#)

4.12 PVH223Al3Config Class Reference

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

Inheritance diagram for PVH223Al3Config::



Public Methods

- [PVH223AlIndex IsA \(\) const](#)

Data Fields

- uint32 [iControlFieldOctets](#)
- uint32 [iSendBufferSize](#)

4.12.1 Detailed Description

PVH223Al3Config class

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

4.12.2 Member Function Documentation

4.12.2.1 [PVH223AlIndex](#) PVH223Al3Config::IsA () const [inline, virtual]

Implements [PVH223AlConfig](#).

4.12.3 Field Documentation

4.12.3.1 uint32 PVH223Al3Config::iControlFieldOctets

4.12.3.2 uint32 PVH223Al3Config::iSendBufferSize

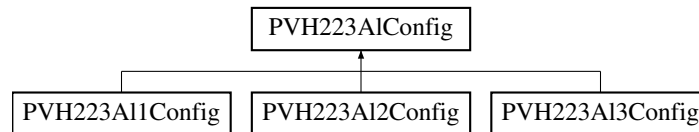
The documentation for this class was generated from the following file:

- [pv_2way_h324m_types.h](#)

4.13 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 Methods

- virtual [PVH223AlIndex](#) [IsA](#) () const=0

4.13.1 Detailed Description

PVH223AlConfig class

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

4.13.2 Member Enumeration Documentation

4.13.2.1 enum PVH223AlConfig::PVH223AlIndex

Enumeration values:

[PVH223_AL1](#)

[PVH223_AL2](#)

[PVH223_AL3](#)

4.13.3 Member Function Documentation

4.13.3.1 virtual [PVH223AlIndex](#) PVH223AlConfig::IsA () [pure virtual]

Implemented in [PVH223Al1Config](#), [PVH223Al2Config](#), and [PVH223Al3Config](#).

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

- [pv_2way_h324m_types.h](#)

Chapter 5

pv2way_engine File Documentation

5.1 pv_2way_basic_types.h File Reference

```
#include "pvmf_format_type.h"
#include "oscl_vector.h"
#include "oscl_mem.h"
#include "pv_2way_codecspecifier_interface.h"
```

Data Structures

- class [PV2WayConnectOptions](#)
- class [PV2WayInitInfo](#)

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 PV2WayErrorRejected = PV2WayErrorStatusStart - 5
- const int PV2WayErrReplaced = PV2WayErrorStatusStart - 6

5.1.1 Typedef Documentation

5.1.1.1 typedef enum TPVDirection PV2WayDirection

TPVDirection Enum

TPVDirection enumerates the direction of the track.

5.1.1.2 typedef enum TPVLoopbackMode PV2WayLoopbackMode

TPVLoopbackMode Enum

TPVLoopbackMode enumerates 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.

Enumeration values:

- Idle** 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 enumerates the direction of the track.

Enumeration values:

PV_DIRECTION_NONE
INCOMING
OUTGOING
PV_DIRECTION_BOTH

5.1.2.3 enum TPVLoopbackMode

TPVLoopbackMode Enum

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

Enumeration values:

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

Enumeration values:

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

Enumeration values:

PV_323
PV_324M
PV_SIP
PV_TERMINAL_TYPE_NONE

5.1.2.6 enum TPVTIndicationType

TPVTIndicationType enum

Enumeration of unsolicited indications from pv2way.

Enumeration values:

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 the remote terminal has paused an incoming track. Any media sink associated with this will be stopped.

PVT_INDICATION_RESUME_TRACK Indicates that the remote terminal has resumed an incoming track. Any media sink associated with this will be restarted.

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 PV2WayDispatchError = PV2WayErrorStatusStart - 1`

There was an error dispatching muxed data to the downstream node *

5.1.3.2 `const int PV2WayErrorRejected = PV2WayErrorStatusStart - 5`

The request was rejected by the peer *

5.1.3.3 `const int PV2WayErrorStatusStart = (-10500)`

The starting error code for 2way specific errors *

5.1.3.4 `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_interface.h File Reference

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

Data Structures

- class [H324MConfigInterface](#)

Defines

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

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](#) }

5.3.1 Define Documentation

- 5.3.1.1** #define [PVH324MConfigUuid](#) PVU-
uid(0x2b0b54e2,0x7079,0x46c6,0xb2,0x3e,0x04,0xff,0xd3,0x0e,0x14,0x36)

5.3.2 Enumeration Type Documentation

5.3.2.1 enum [PVH324MIndicationType](#)

[PVH324MIndicationType](#) enum

Enumeration of unsolicited H324m specific indications from pv2way.

Enumeration values:

[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 local buffer contains the indices of the user input formats supported by the peer.
- PV_INDICATION_USER_INPUT** Indicates the receipt of user input from the remote terminal. The derived class contains 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.

5.4 pv_2way_h324m_types.h File Reference

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

Data Structures

- class [PV2Way324ConnectOptions](#)
- class [PV2Way324InitInfo](#)
- class [PVH223A11Config](#)
- class [PVH223A12Config](#)
- class [PVH223A13Config](#)
- class [PVH223A1Config](#)

Defines

- #define [PV_2WAY_MAX_USER_INPUT_FORMATS](#) 4
- #define [PV_2WAY_MAX_SKEW_MS](#) 1000

Typedefs

- typedef enum [TPVPostDisconnectOption](#) [PV2WayPostDisconnectOption](#)
- typedef enum [TPVUserInputType](#) [PV2WayUserInputType](#)

Enumerations

- enum [TPVPostDisconnectOption](#) { [EDisconnectLine](#), [EAnalogueTelephony](#) }
- enum [TPVUserInputType](#) { [EAlphanumeric](#) = 0, [EDtmf](#) }

5.4.1 Define Documentation

5.4.1.1 #define [PV_2WAY_MAX_SKEW_MS](#) 1000

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

5.4.1.2 #define [PV_2WAY_MAX_USER_INPUT_FORMATS](#) 4

The maximum number of supported formats for user input *

5.4.2 Typedef Documentation

5.4.2.1 typedef enum [TPVPostDisconnectOption](#) [PV2WayPostDisconnectOption](#)

TPVPostDisconnectOption Enum

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

5.4.2.2 typedef enum **TPVUserInputType** PV2WayUserInputType

TPVUserInputType enum Enumeration of user input types

5.4.3 Enumeration Type Documentation

5.4.3.1 enum TPVPostDisconnectOption

TPVPostDisconnectOption Enum

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

Enumeration values:

EDisconnectLine

EAnalogueTelephony

5.4.3.2 enum TPVUserInputType

TPVUserInputType enum Enumeration of user input types

Enumeration values:

EAlphanumeric

EDtmf

5.5 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](#)

5.6 pv_2way_proxy_factory.h File Reference

```
#include "pv_common_types.h"
#include "pv_2way_interface.h"
#include "pv_engine_observer.h"
```

Data Structures

- class [CPV2WayProxyFactory](#)

5.7 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, 0xe3)

5.7.1 Define Documentation

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

Index

- ~CPV2WayInterface
 - CPV2WayInterface, [7](#)
- ~PV2Way324ConnectOptions
 - PV2Way324ConnectOptions, [25](#)
- ~PV2Way324InitInfo
 - PV2Way324InitInfo, [26](#)
- ~PV2WayInitInfo
 - PV2WayInitInfo, [30](#)
- AcceptableFormatsMatch
 - PV2WayTestExtensionInterface, [32](#)
- AddDataSink
 - CPV2WayInterface, [7](#)
- AddDataSource
 - CPV2WayInterface, [7](#)
- addRef
 - PV2WayTestExtensionInterface, [32](#)
- CancelAllCommands
 - CPV2WayInterface, [8](#)
- Cleanup
 - CPV2WayEngineFactory, [4](#)
 - CPV2WayProxyFactory, [15](#)
- Connect
 - CPV2WayInterface, [8](#)
- CPV2WayEngineFactory, [4](#)
- CPV2WayEngineFactory
 - Cleanup, [4](#)
 - CreateTerminal, [4](#)
 - DeleteTerminal, [4](#)
 - Init, [5](#)
- CPV2WayInterface, [6](#)
- CPV2WayInterface
 - ~CPV2WayInterface, [7](#)
 - AddDataSink, [7](#)
 - AddDataSource, [7](#)
 - CancelAllCommands, [8](#)
 - Connect, [8](#)
 - Disconnect, [8](#)
 - GetLogLevel, [9](#)
 - GetSDKInfo, [9](#)
 - GetSDKModuleInfo, [9](#)
 - GetState, [10](#)
 - Init, [10](#)
 - Pause, [10](#)
 - QueryInterface, [11](#)
 - QueryUUID, [11](#)
 - RemoveDataSink, [11](#)
 - RemoveDataSource, [12](#)
 - RemoveLogAppender, [12](#)
 - Reset, [12](#)
 - Resume, [13](#)
 - SetLogAppender, [13](#)
 - SetLogLevel, [13](#)
- CPV2WayProxyFactory, [15](#)
- CPV2WayProxyFactory
 - Cleanup, [15](#)
 - CreateTerminal, [15](#)
 - DeleteTerminal, [15](#)
 - Init, [15](#)
- CreateTerminal
 - CPV2WayEngineFactory, [4](#)
 - CPV2WayProxyFactory, [15](#)
- DeleteTerminal
 - CPV2WayEngineFactory, [4](#)
 - CPV2WayProxyFactory, [15](#)
- Disconnect
 - CPV2WayInterface, [8](#)
- EAlphanumeric
 - pv_2way_h324m_types.h, [46](#)
- EAnalogueTelephony
 - pv_2way_h324m_types.h, [46](#)
- EConnected
 - pv_2way_basic_types.h, [39](#)
- EConnecting
 - pv_2way_basic_types.h, [39](#)
- EDisconnecting
 - pv_2way_basic_types.h, [39](#)
- EDisconnectLine
 - pv_2way_h324m_types.h, [46](#)
- EDtmf
 - pv_2way_h324m_types.h, [46](#)
- EIdle
 - pv_2way_basic_types.h, [38](#)
- EInitializing
 - pv_2way_basic_types.h, [38](#)
- EResetting
 - pv_2way_basic_types.h, [39](#)

- ESetup
 - pv_2way_basic_types.h, 39
- FastUpdate
 - H324MConfigInterface, 17
- GetConnectInfoClassName
 - PV2Way324ConnectOptions, 25
 - PV2WayConnectOptions, 29
- GetInitInfoClassName
 - PV2Way324InitInfo, 26
 - PV2WayInitInfo, 30
- GetLogLevel
 - CPV2WayInterface, 9
- GetSDKInfo
 - CPV2WayInterface, 9
- GetSDKModuleInfo
 - CPV2WayInterface, 9
- GetState
 - CPV2WayInterface, 10
- H324MConfigInterface, 16
- H324MConfigInterface
 - FastUpdate, 17
 - SendEndSession, 17
 - SendRme, 17
 - SendRtd, 17
 - SendSkewIndication, 18
 - SendUserInput, 18
 - SendVendorId, 18
 - SendVideoTemporalSpatialTradeoffCommand, 18
 - SendVideoTemporalSpatialTradeoffIndication, 18
 - SetEndSessionTimeout, 19
 - SetIncomingChannelConfiguration, 19
 - SetLogicalChannelBufferingMs, 19
 - SetMaxMuxCcslSduSize, 19
 - SetMaxMuxPduSize, 20
 - SetMaxPduSize, 20
 - SetMaxSduSize, 20
 - SetMaxSduSizeR, 20
 - SetMultiplexLevel, 21
 - SetObservers, 21
 - SetOutgoingChannelConfiguration, 21
 - SetTerminalType, 21
 - SetTimerCounter, 22
 - SetVendor, 22
 - SetVideoResolutions, 23
- iControlFieldOctets
 - PVH223Al3Config, 35
- iDisconnectTimeoutInterval
 - PV2Way324ConnectOptions, 25
- iFramed
 - PVH223Al1Config, 33
- iIncomingAudioFormats
 - PV2WayInitInfo, 30
- iIncomingVideoFormats
 - PV2WayInitInfo, 30
- iLocalId
 - PV2WayConnectOptions, 29
- iLocalIdSize
 - PV2WayConnectOptions, 29
- iLoopbackMode
 - PV2WayConnectOptions, 29
- iMultiplexingDelayMs
 - PV2Way324InitInfo, 27
- INCOMING
 - pv_2way_basic_types.h, 39
- Init
 - CPV2WayEngineFactory, 5
 - CPV2WayInterface, 10
 - CPV2WayProxyFactory, 15
- iOutgoingAudioFormats
 - PV2WayInitInfo, 31
- iOutgoingVideoFormats
 - PV2WayInitInfo, 31
- iRemoteId
 - PV2WayConnectOptions, 29
- iRemoteIdSize
 - PV2WayConnectOptions, 29
- IsA
 - PVH223Al1Config, 33
 - PVH223Al2Config, 34
 - PVH223Al3Config, 35
 - PVH223AlConfig, 36
- iSendBufferSize
 - PVH223Al3Config, 35
- iUseSequenceNumbers
 - PVH223Al2Config, 34
- OUTGOING
 - pv_2way_basic_types.h, 39
- Pause
 - CPV2WayInterface, 10
- PV2Way324ConnectOptions, 24
 - PV2Way324ConnectOptions, 24
- PV2Way324ConnectOptions
 - ~PV2Way324ConnectOptions, 25
 - GetConnectInfoClassName, 25
 - iDisconnectTimeoutInterval, 25
 - PV2Way324ConnectOptions, 24
- PV2Way324InitInfo, 26
 - PV2Way324InitInfo, 26
- PV2Way324InitInfo
 - ~PV2Way324InitInfo, 26

- GetInitInfoClassName, 26
- iMultiplexingDelayMs, 27
- PV2Way324InitInfo, 26
- PV2WayConnectOptions, 28
 - PV2WayConnectOptions, 28
- PV2WayConnectOptions
 - GetConnectInfoClassName, 29
 - iLocalId, 29
 - iLocalIdSize, 29
 - iLoopbackMode, 29
 - iRemoteId, 29
 - iRemoteIdSize, 29
 - PV2WayConnectOptions, 28
- PV2WayDirection
 - pv_2way_basic_types.h, 38
- PV2WayDispatchError
 - pv_2way_basic_types.h, 41
- PV2WayErrorRejected
 - pv_2way_basic_types.h, 41
- PV2WayErrorStatusStart
 - pv_2way_basic_types.h, 41
- PV2WayErrReplaced
 - pv_2way_basic_types.h, 41
- PV2WayInitInfo, 30
- PV2WayInitInfo
 - ~PV2WayInitInfo, 30
 - GetInitInfoClassName, 30
 - iIncomingAudioFormats, 30
 - iIncomingVideoFormats, 30
 - iOutgoingAudioFormats, 31
 - iOutgoingVideoFormats, 31
- PV2WayLoopbackMode
 - pv_2way_basic_types.h, 38
- PV2WayMediaType
 - pv_2way_basic_types.h, 38
- PV2WayPostDisconnectOption
 - pv_2way_h324m_types.h, 45
- PV2WayState
 - pv_2way_basic_types.h, 38
- PV2WayTerminalType
 - pv_2way_basic_types.h, 38
- PV2WayTestEncExtensionUUID
 - pv_2way_test_extension_interface.h, 49
- PV2WayTestExtensionInterface, 32
- PV2WayTestExtensionInterface
 - AcceptableFormatsMatch, 32
 - addRef, 32
 - queryInterface, 32
 - removeRef, 32
- PV2WayUserInputType
 - pv_2way_h324m_types.h, 45
- pv_2way_basic_types.h
 - EConnected, 39
 - EConnecting, 39
 - EDisconnecting, 39
 - EIdle, 38
 - EInitializing, 38
 - EResetting, 39
 - ESetup, 39
 - INCOMING, 39
 - OUTGOING, 39
 - PV_323, 40
 - PV_324M, 40
 - PV_AUDIO, 40
 - PV_CONTROL, 40
 - PV_DATA, 40
 - PV_DIRECTION_BOTH, 39
 - PV_DIRECTION_NONE, 39
 - PV_LOOPBACK_COMM, 39
 - PV_LOOPBACK_ENGINE, 39
 - PV_LOOPBACK_MUX, 39
 - PV_LOOPBACK_NONE, 39
 - PV_MEDIA_ALL, 40
 - PV_MEDIA_NONE, 40
 - PV_MULTIPLEXED, 40
 - PV_SIP, 40
 - PV_TERMINAL_TYPE_NONE, 40
 - PV_USER_INPUT, 40
 - PV_VIDEO, 40
 - PVT_INDICATION_CLOSE_TRACK, 40
 - PVT_INDICATION_CLOSING_TRACK, 40
 - PVT_INDICATION_DISCONNECT, 40
 - PVT_INDICATION_INCOMING_TRACK, 40
 - PVT_INDICATION_INTERNAL_ERROR, 41
 - PVT_INDICATION_OUTGOING_TRACK, 40
 - PVT_INDICATION_PAUSE_TRACK, 40
 - PVT_INDICATION_RESUME_TRACK, 41
- pv_2way_basic_types.h, 37
 - PV2WayDirection, 38
 - PV2WayDispatchError, 41
 - PV2WayErrorRejected, 41
 - PV2WayErrorStatusStart, 41
 - PV2WayErrReplaced, 41
 - PV2WayLoopbackMode, 38
 - PV2WayMediaType, 38
 - PV2WayState, 38
 - PV2WayTerminalType, 38
 - PVTrackId, 38
 - TPVDirection, 39
 - TPVLoopbackMode, 39
 - TPVMediaType_t, 39
 - TPVTerminalType, 40
 - TPVTIndicationType, 40

- pv_2way_engine_factory.h, [42](#)
- pv_2way_h324m_interface.h
 - PV_INDICATION_FAST_UPDATE, [43](#)
 - PV_INDICATION_RME, [43](#)
 - PV_INDICATION_RTD, [43](#)
 - PV_INDICATION_SKEW, [44](#)
 - PV_INDICATION_USER_INPUT, [44](#)
 - PV_INDICATION_USER_INPUT_-CAPABILITY, [44](#)
 - PV_INDICATION_VENDOR_ID, [44](#)
 - PV_INDICATION_VIDEO_SPATIAL_-TEMPORAL_TRADEOFF_-COMMAND, [43](#)
 - PV_INDICATION_VIDEO_SPATIAL_-TEMPORAL_TRADEOFF_-INDICATION, [43](#)
- pv_2way_h324m_interface.h, [43](#)
 - PVH324MConfigUuid, [43](#)
 - PVH324MIndicationType, [43](#)
- pv_2way_h324m_types.h
 - EAlphanumeric, [46](#)
 - EAnalogueTelephony, [46](#)
 - EDisconnectLine, [46](#)
 - EDtmf, [46](#)
- pv_2way_h324m_types.h, [45](#)
 - PV2WayPostDisconnectOption, [45](#)
 - PV2WayUserInputType, [45](#)
 - PV_2WAY_MAX_SKEW_MS, [45](#)
 - PV_2WAY_MAX_USER_INPUT_-FORMATS, [45](#)
 - TPVPostDisconnectOption, [46](#)
 - TPVUserInputType, [46](#)
- pv_2way_interface.h, [47](#)
- PV_2WAY_MAX_SKEW_MS
 - pv_2way_h324m_types.h, [45](#)
- PV_2WAY_MAX_USER_INPUT_FORMATS
 - pv_2way_h324m_types.h, [45](#)
- pv_2way_proxy_factory.h, [48](#)
- pv_2way_test_extension_interface.h, [49](#)
 - PV2WayTestEncExtensionUUID, [49](#)
- PV_323
 - pv_2way_basic_types.h, [40](#)
- PV_324M
 - pv_2way_basic_types.h, [40](#)
- PV_AUDIO
 - pv_2way_basic_types.h, [40](#)
- PV_CONTROL
 - pv_2way_basic_types.h, [40](#)
- PV_DATA
 - pv_2way_basic_types.h, [40](#)
- PV_DIRECTION_BOTH
 - pv_2way_basic_types.h, [39](#)
- PV_DIRECTION_NONE
 - pv_2way_basic_types.h, [39](#)
- PV_INDICATION_FAST_UPDATE
 - pv_2way_h324m_interface.h, [43](#)
- PV_INDICATION_RME
 - pv_2way_h324m_interface.h, [43](#)
- PV_INDICATION_RTD
 - pv_2way_h324m_interface.h, [43](#)
- PV_INDICATION_SKEW
 - pv_2way_h324m_interface.h, [44](#)
- PV_INDICATION_USER_INPUT
 - pv_2way_h324m_interface.h, [44](#)
- PV_INDICATION_USER_INPUT_-CAPABILITY
 - pv_2way_h324m_interface.h, [44](#)
- PV_INDICATION_VENDOR_ID
 - pv_2way_h324m_interface.h, [44](#)
- PV_INDICATION_VIDEO_SPATIAL_-TEMPORAL_TRADEOFF_-COMMAND
 - pv_2way_h324m_interface.h, [43](#)
- PV_INDICATION_VIDEO_SPATIAL_-TEMPORAL_TRADEOFF_-INDICATION
 - pv_2way_h324m_interface.h, [43](#)
- PV_LOOPBACK_COMM
 - pv_2way_basic_types.h, [39](#)
- PV_LOOPBACK_ENGINE
 - pv_2way_basic_types.h, [39](#)
- PV_LOOPBACK_MUX
 - pv_2way_basic_types.h, [39](#)
- PV_LOOPBACK_NONE
 - pv_2way_basic_types.h, [39](#)
- PV_MEDIA_ALL
 - pv_2way_basic_types.h, [40](#)
- PV_MEDIA_NONE
 - pv_2way_basic_types.h, [40](#)
- PV_MULTIPLEXED
 - pv_2way_basic_types.h, [40](#)
- PV_SIP
 - pv_2way_basic_types.h, [40](#)
- PV_TERMINAL_TYPE_NONE
 - pv_2way_basic_types.h, [40](#)
- PV_USER_INPUT
 - pv_2way_basic_types.h, [40](#)
- PV_VIDEO
 - pv_2way_basic_types.h, [40](#)
- PVH223_AL1
 - PVH223AlConfig, [36](#)
- PVH223_AL2
 - PVH223AlConfig, [36](#)
- PVH223_AL3
 - PVH223AlConfig, [36](#)
- PVH223Al1Config, [33](#)
 - iFramed, [33](#)
 - IsA, [33](#)

- PVH223AI2Config, 34
 - IsA, 34
 - iUseSequenceNumbers, 34
- PVH223AI3Config, 35
 - iControlFieldOctets, 35
 - IsA, 35
 - iSendBufferSize, 35
- PVH223AIConfig, 36
 - PVH223_AL1, 36
 - PVH223_AL2, 36
 - PVH223_AL3, 36
- PVH223AIConfig
 - IsA, 36
 - PVH223AIIndex, 36
- PVH223AIIndex
 - PVH223AIConfig, 36
- PVH324MConfigUid
 - pv_2way_h324m_interface.h, 43
- PVH324MIndicationType
 - pv_2way_h324m_interface.h, 43
- PVT_INDICATION_CLOSE_TRACK
 - pv_2way_basic_types.h, 40
- PVT_INDICATION_CLOSING_TRACK
 - pv_2way_basic_types.h, 40
- PVT_INDICATION_DISCONNECT
 - pv_2way_basic_types.h, 40
- PVT_INDICATION_INCOMING_TRACK
 - pv_2way_basic_types.h, 40
- PVT_INDICATION_INTERNAL_ERROR
 - pv_2way_basic_types.h, 41
- PVT_INDICATION_OUTGOING_TRACK
 - pv_2way_basic_types.h, 40
- PVT_INDICATION_PAUSE_TRACK
 - pv_2way_basic_types.h, 40
- PVT_INDICATION_RESUME_TRACK
 - pv_2way_basic_types.h, 41
- PVTrackId
 - pv_2way_basic_types.h, 38
- QueryInterface
 - CPV2WayInterface, 11
- queryInterface
 - PV2WayTestExtensionInterface, 32
- QueryUUID
 - CPV2WayInterface, 11
- RemoveDataSink
 - CPV2WayInterface, 11
- RemoveDataSource
 - CPV2WayInterface, 12
- RemoveLogAppender
 - CPV2WayInterface, 12
- removeRef
 - PV2WayTestExtensionInterface, 32
- Reset
 - CPV2WayInterface, 12
- Resume
 - CPV2WayInterface, 13
- SendEndSession
 - H324MConfigInterface, 17
- SendRme
 - H324MConfigInterface, 17
- SendRtd
 - H324MConfigInterface, 17
- SendSkewIndication
 - H324MConfigInterface, 18
- SendUserInput
 - H324MConfigInterface, 18
- SendVendorId
 - H324MConfigInterface, 18
- SendVideoTemporalSpatialTradeoffCommand
 - H324MConfigInterface, 18
- SendVideoTemporalSpatialTradeoffIndication
 - H324MConfigInterface, 18
- SetEndSessionTimeout
 - H324MConfigInterface, 19
- SetIncomingChannelConfiguration
 - H324MConfigInterface, 19
- SetLogAppender
 - CPV2WayInterface, 13
- SetLogicalChannelBufferingMs
 - H324MConfigInterface, 19
- SetLogLevel
 - CPV2WayInterface, 13
- SetMaxMuxCcsrcsduSize
 - H324MConfigInterface, 19
- SetMaxMuxPduSize
 - H324MConfigInterface, 20
- SetMaxPduSize
 - H324MConfigInterface, 20
- SetMaxSduSize
 - H324MConfigInterface, 20
- SetMaxSduSizeR
 - H324MConfigInterface, 20
- SetMultiplexLevel
 - H324MConfigInterface, 21
- SetObservers
 - H324MConfigInterface, 21
- SetOutgoingChannelConfiguration
 - H324MConfigInterface, 21
- SetTerminalType
 - H324MConfigInterface, 21
- SetTimerCounter
 - H324MConfigInterface, 22
- SetVendor
 - H324MConfigInterface, 22
- SetVideoResolutions

H324MConfigInterface, [23](#)

TPVDirection

pv_2way_basic_types.h, [39](#)

TPVLoopbackMode

pv_2way_basic_types.h, [39](#)

TPVMediaType_t

pv_2way_basic_types.h, [39](#)

TPVPostDisconnectOption

pv_2way_h324m_types.h, [46](#)

TPVTerminalType

pv_2way_basic_types.h, [40](#)

TPVTIndicationType

pv_2way_basic_types.h, [40](#)

TPVUserInputType

pv_2way_h324m_types.h, [46](#)