



packetvideo™

PV2Way Engine

Build Version: CORE\_8.000.1.1\_RC4

November 1, 2009

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>pv2way_engine Hierarchical Index</b>                            | <b>1</b>  |
| 1.1      | pv2way_engine Class Hierarchy . . . . .                            | 1         |
| <b>2</b> | <b>pv2way_engine Data Structure Index</b>                          | <b>2</b>  |
| 2.1      | pv2way_engine Data Structures . . . . .                            | 2         |
| <b>3</b> | <b>pv2way_engine File Index</b>                                    | <b>3</b>  |
| 3.1      | pv2way_engine File List . . . . .                                  | 3         |
| <b>4</b> | <b>pv2way_engine Data Structure Documentation</b>                  | <b>4</b>  |
| 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      | H324MConfigObserver Class Reference . . . . .                      | 25        |
| 4.6      | H324MReverseParametersExtensionInterface Class Reference . . . . . | 26        |
| 4.7      | PV2Way324ConnectOptions Class Reference . . . . .                  | 27        |
| 4.8      | PV2Way324InitInfo Class Reference . . . . .                        | 29        |
| 4.9      | PV2WayConnectOptions Class Reference . . . . .                     | 31        |
| 4.10     | PV2WayInitInfo Class Reference . . . . .                           | 33        |
| 4.11     | PV2WayTestExtensionInterface Class Reference . . . . .             | 35        |
| 4.12     | PVH223A11Config Class Reference . . . . .                          | 36        |
| 4.13     | PVH223A12Config Class Reference . . . . .                          | 37        |
| 4.14     | PVH223A13Config Class Reference . . . . .                          | 38        |
| 4.15     | PVH223A1Config Class Reference . . . . .                           | 39        |
| 4.16     | PVMFCComponentInterface Class Reference . . . . .                  | 40        |
| <b>5</b> | <b>pv2way_engine File Documentation</b>                            | <b>41</b> |
| 5.1      | pv_2way_basic_types.h File Reference . . . . .                     | 41        |
| 5.2      | pv_2way_engine_factory.h File Reference . . . . .                  | 46        |

---

|     |   |    |
|-----|---|----|
| 5.3 | <a href="#">pv_2way_h324m_types.h File Reference</a>              | 47 |
| 5.4 | <a href="#">pv_2way_interface.h File Reference</a>                | 49 |
| 5.5 | <a href="#">pv_2way_proxy_factory.h File Reference</a>            | 50 |
| 5.6 | <a href="#">pv_2way_test_extension_interface.h File Reference</a> | 51 |
| 5.7 | <a href="#">tsc_h324m_config_interface.h File Reference</a>       | 52 |

# 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 |
| H324MConfigObserver . . . . .                      | 25 |
| H324MReverseParametersExtensionInterface . . . . . | 26 |
| PV2WayConnectOptions . . . . .                     | 31 |
| PV2Way324ConnectOptions . . . . .                  | 27 |
| PV2WayInitInfo . . . . .                           | 33 |
| PV2Way324InitInfo . . . . .                        | 29 |
| PV2WayTestExtensionInterface . . . . .             | 35 |
| PVH223AIConfig . . . . .                           | 39 |
| PVH223AI1Config . . . . .                          | 36 |
| PVH223AI2Config . . . . .                          | 37 |
| PVH223AI3Config . . . . .                          | 38 |
| PVMFComponentInterface . . . . .                   | 40 |

## 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 |
| H324MConfigObserver                      | 25 |
| H324MReverseParametersExtensionInterface | 26 |
| PV2Way324ConnectOptions                  | 27 |
| PV2Way324InitInfo                        | 29 |
| PV2WayConnectOptions                     | 31 |
| PV2WayInitInfo                           | 33 |
| PV2WayTestExtensionInterface             | 35 |
| PVH223A11Config                          | 36 |
| PVH223A12Config                          | 37 |
| PVH223A13Config                          | 38 |
| PVH223A1Config                           | 39 |
| PVMFComponentInterface                   | 40 |

# Chapter 3

## pv2way\_engine File Index

### 3.1 pv2way\_engine File List

Here is a list of all files with brief descriptions:

|  |    |
|--|----|
| <a href="#">pv_2way_basic_types.h</a>              | 41 |
| <a href="#">pv_2way_engine_factory.h</a>           | 46 |
| <a href="#">pv_2way_h324m_types.h</a>              | 47 |
| <a href="#">pv_2way_interface.h</a>                | 49 |
| <a href="#">pv_2way_proxy_factory.h</a>            | 50 |
| <a href="#">pv_2way_test_extension_interface.h</a> | 51 |
| <a href="#">tsc_h324m_config_interface.h</a>       | 52 |

# Chapter 4

## pv2way\_engine Data Structure Documentation

### 4.1 CPV2WayEngineFactory Class Reference

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

#### Static Public Methods

- OSCL\_IMPORT\_REF void [Init](#) ()
- OSCL\_IMPORT\_REF void [Cleanup](#) ()
- OSCL\_IMPORT\_REF [CPV2WayInterface](#) \* [CreateTerminal](#) ([PV2WayTerminalType](#) aTerminalType, PVCommandStatusObserver \*aCmdStatusObserver, PVInformationalEventObserver \*aInfoEventObserver, PVErrrorEventObserver \*aErrorEventObserver)
- OSCL\_IMPORT\_REF void [DeleteTerminal](#) ([CPV2WayInterface](#) \*terminal)

#### 4.1.1 Member Function Documentation

4.1.1.1 OSCL\_IMPORT\_REF void [CPV2WayEngineFactory::Cleanup](#) () [static]

4.1.1.2 OSCL\_IMPORT\_REF [CPV2WayInterface](#)\* [CPV2WayEngineFactory::CreateTerminal](#) ([PV2WayTerminalType](#) 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

**4.1.1.3 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 PVAyncInformationalEvent. 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 PVAyncInformationalEvent. 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 (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  
 @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 PvmfMimeType & 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 (PVMFNNodeInterface & 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::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

#### 4.2.3.17 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, PVErrorEventObserver \*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, 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 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 <tsc_h324m_config_interface.h>
```

### Public Methods

- virtual void [SetObserver](#) (H324MConfigObserver \*aObserver)=0
- virtual PVMFCCommandId [SetMultiplexLevel](#) (TPVH223Level aLevel, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SetMaxSduSize](#) (TPVAdaptationLayer aLayer, int32 aSize, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SetMaxSduSizeR](#) (TPVAdaptationLayer aLayer, int32 aSize, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [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 PVMFCCommandId [SetA12SequenceNumbers](#) (int32 aSeqNumWidth, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SetA13ControlFieldOctets](#) (int32 aCfo, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SetMaxPduSize](#) (int32 aMaxPduSize, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SetTerminalType](#) (uint8 aTerminalType, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SetALConfiguration](#) (TPVMediaType\_t aMediaType, TPVAdaptationLayer aLayer, bool aAllow, bool aUse=true, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SendRme](#) (OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SetMaxMuxPduSize](#) (int32 aRequestMaxMuxPduSize, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SetMaxMuxCcsrlSduSize](#) (int32 aMaxCcsrlSduSize, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [FastUpdate](#) (PVMFNodeInterface &aTrack, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SendRtd](#) (OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SetVendor](#) (uint8 cc, uint8 ext, uint32 mc, const uint8 \*aProduct, uint16 aProductLen, const uint8 \*aVersion, uint16 aVersionLen, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SendEndSession](#) (OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SetEndSessionTimeout](#) (uint32 aTimeout, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SetTimerCounter](#) (TPVH324TimerCounter aTimerCounter, uint8 aSeries, uint32 aSeriesOffset, uint32 aValue, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SetVideoResolutions](#) (TPVDirection aDirection, Oscl\_Vector< PVMFVideoResolutionRange, OsclMemAllocator > &aResolutions, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SendVendorId](#) (OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SendVideoTemporalSpatialTradeoffCommand](#) (TPVChannelId aLogicalChannel, uint8 aTradeoff, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId [SendVideoTemporalSpatialTradeoffIndication](#) (TPVChannelId aLogicalChannel, uint8 aTradeoff, OsclAny \*aContextData=NULL)=0

- virtual PVMFCCommandId `SendSkewIndication` (TPVChannelId aLogicalChannel1, TPVChannelId aLogicalChannel2, uint16 aSkew, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId `SetLogicalChannelBufferingMs` (uint32 aInBufferingMs, uint32 aOutBufferingMs, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId `SendUserInput` (CPVUserInput \*user\_input, OsclAny \*aContextData=NULL)=0
- virtual PVMFCCommandId `SetWnsrp` (const bool aEnableWnsrp, OsclAny \*aContextData=NULL)=0

### 4.4.1 Member Function Documentation

#### 4.4.1.1 virtual PVMFCCommandId 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

#### 4.4.1.2 virtual PVMFCCommandId 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

#### 4.4.1.3 virtual PVMFCCommandId 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

**4.4.1.4 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

**4.4.1.5 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.

**4.4.1.6 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 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

**4.4.1.7 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.

**4.4.1.8 virtual PVMFCommandId H324MConfigInterface::SendVideoTemporalSpatialTradeoff-Command (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.

**4.4.1.9 virtual PVMFCommandId H324MConfigInterface::SendVideoTemporalSpatialTradeoff-Indication (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.

**4.4.1.10 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

**4.4.1.11 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

**4.4.1.12 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

**4.4.1.13** `virtual PVMFCommandId H324MConfigInterface::SetCodecPreference (Osci_Vector< PVMFFormatType, OsciMemAllocator > & aIncomingAudio, Osci_Vector< PVMFFormatType, OsciMemAllocator > & aIncomingVideo, Osci_Vector< PVMFFormatType, OsciMemAllocator > & aOutGoingAudio, Osci_Vector< PVMFFormatType, OsciMemAllocator > & aOutGoingVideo, OsciAny * 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

**4.4.1.14** `virtual PVMFCommandId H324MConfigInterface::SetEndSessionTimeout (uint32 aTimeout, OsciAny * 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

**4.4.1.15** `virtual PVMFCommandId H324MConfigInterface::SetLogicalChannelBufferingMs (uint32 aInBufferingMs, uint32 aOutBufferingMs, OsciAny * 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

**4.4.1.16 virtual PVMFCommandId H324MConfigInterface::SetMaxMuxCcslSduSize (int32 *aMaxCcslSduSize*, OsclAny \* *aContextData* = NULL) [pure virtual]**

This API sets the max ccsl sdu size

**Parameters:**

*aMaxCcslSduSize* The max ccsl 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

**4.4.1.17 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

**4.4.1.18 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

**4.4.1.19 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



**4.4.1.20** `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

**4.4.1.21** `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 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

**4.4.1.22** `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

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

This API allows the user to specify the terminal type that is advertised 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

### 4.4.1.24 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

### 4.4.1.25 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 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

### 4.4.1.26 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

**4.4.1.27 virtual PVMFCommandId H324MConfigInterface::SetWnsrp (const bool *aEnableWnsrp*, OsciAny \* *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](#)

## 4.5 H324MConfigObserver Class Reference

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

### Public Methods

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

### 4.5.1 Constructor & Destructor Documentation

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

### 4.5.2 Member Function Documentation

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

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

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

- [tsc\\_h324m\\_config\\_interface.h](#)

## 4.6 H324MReverseParametersExtensionInterface Class Reference

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

### Public Methods

- virtual const PvmfMimeString \* [GetFormatCapabilities](#) ()=0
- virtual void [SetPortTag](#) (int32 aPortTag)=0
- virtual int32 [GetPortTag](#) () const=0

### 4.6.1 Detailed Description

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

### 4.6.2 Member Function Documentation

**4.6.2.1** virtual const PvmfMimeString\* [H324MReverseParametersExtensionInterface::GetFormatCapabilities](#) () [pure virtual]

**Returns:**

The reverse media format and capabilities.

**4.6.2.2** virtual int32 [H324MReverseParametersExtensionInterface::GetPortTag](#) () [pure virtual]

**Returns:**

The port tag for the reverse channel

**4.6.2.3** virtual void [H324MReverseParametersExtensionInterface::SetPortTag](#) (int32 *aPortTag*) [pure virtual]

**Parameters:**

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

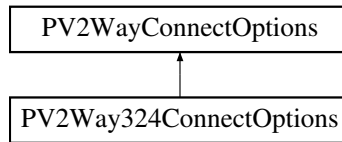
The documentation for this class was generated from the following file:

- [tsc\\_h324m\\_config\\_interface.h](#)

## 4.7 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](#)

#### 4.7.1 Detailed Description

PV2Way324ConnectOptions Class

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

#### 4.7.2 Constructor & Destructor Documentation

##### 4.7.2.1 PV2Way324ConnectOptions::PV2Way324ConnectOptions (uint32 aDisconnectTimeoutInterval) [inline]

Constructor

##### Parameters:

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

**4.7.2.2** PV2Way324ConnectOptions::PV2Way324ConnectOptions () [inline]

**4.7.2.3** virtual PV2Way324ConnectOptions::~~PV2Way324ConnectOptions () [inline, virtual]

### 4.7.3 Member Function Documentation

**4.7.3.1** 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](#).

### 4.7.4 Field Documentation

**4.7.4.1** uint32 PV2Way324ConnectOptions::iDisconnectTimeoutInterval

The disconnect timeout interval in units of 100ms

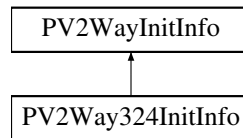
The documentation for this class was generated from the following file:

- [pv\\_2way\\_h324m\\_types.h](#)

## 4.8 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](#)

### 4.8.1 Detailed Description

PV2Way324InitInfo Class

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

### 4.8.2 Constructor & Destructor Documentation

**4.8.2.1** [PV2Way324InitInfo::PV2Way324InitInfo](#) () [inline]

**4.8.2.2** [virtual PV2Way324InitInfo::~~PV2Way324InitInfo](#) () [inline, virtual]

### 4.8.3 Member Function Documentation

**4.8.3.1** [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](#).



## 4.8.4 Field Documentation

### 4.8.4.1 uint16 PV2Way324InitInfo::iMultiplexingDelayMs

The Multiplexing delay in Milliseconds

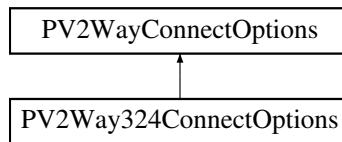
The documentation for this class was generated from the following file:

- [pv\\_2way\\_h324m\\_types.h](#)

## 4.9 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](#)

### 4.9.1 Detailed Description

PV2WayConnectOptions Class

PV2WayConnectOptions class contains options to be specified during connect

### 4.9.2 Constructor & Destructor Documentation

#### 4.9.2.1 PV2WayConnectOptions::PV2WayConnectOptions () [inline]

Default Constructor

#### 4.9.2.2 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

### 4.9.3 Member Function Documentation

#### 4.9.3.1 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](#).

### 4.9.4 Field Documentation

#### 4.9.4.1 uint8\* PV2WayConnectOptions::iLocalId

The id of the local terminal

#### 4.9.4.2 uint32 PV2WayConnectOptions::iLocalIdSize

The size of the local id

#### 4.9.4.3 **TPVLoopbackMode** PV2WayConnectOptions::iLoopbackMode

The loopback mode

#### 4.9.4.4 uint8\* PV2WayConnectOptions::iRemoteId

The id of the peer

#### 4.9.4.5 uint32 PV2WayConnectOptions::iRemoteIdSize

The size of the remote id

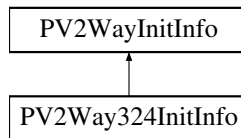
The documentation for this class was generated from the following file:

- [pv\\_2way\\_basic\\_types.h](#)

## 4.10 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

- Oslc\_Vector< PVMFFormatType, OslcMemAllocator > [iOutgoingAudioFormats](#)
- Oslc\_Vector< PVMFFormatType, OslcMemAllocator > [iOutgoingVideoFormats](#)
- Oslc\_Vector< PVMFFormatType, OslcMemAllocator > [iIncomingAudioFormats](#)
- Oslc\_Vector< PVMFFormatType, OslcMemAllocator > [iIncomingVideoFormats](#)

### 4.10.1 Detailed Description

PV2WayInitInfo Class

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

### 4.10.2 Constructor & Destructor Documentation

**4.10.2.1** virtual PV2WayInitInfo::~~PV2WayInitInfo () [inline, virtual]

### 4.10.3 Member Function Documentation

**4.10.3.1** 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](#).

### 4.10.4 Field Documentation

**4.10.4.1** Oslc\_Vector<PVMFFormatType, OslcMemAllocator> PV2WayInitInfo::iIncomingAudioFormats

The list of audio formats that can be received

**4.10.4.2 Osci\_Vector<PVMFFormatType, OsciMemAllocator> PV2WayInitInfo::iIncoming-VideoFormats**

The list of video formats that can be received

**4.10.4.3 Osci\_Vector<PVMFFormatType, OsciMemAllocator> PV2WayInitInfo::iOutgoing-AudioFormats**

The list of audio formats that can be transmitted

**4.10.4.4 Osci\_Vector<PVMFFormatType, OsciMemAllocator> PV2WayInitInfo::iOutgoing-VideoFormats**

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](#)

## 4.11 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 `NegotiatedFormatsMatch (OscL_Vector< FormatCapabilityInfo, OscLMemAllocator > &iInAudFormatCapability, OscL_Vector< FormatCapabilityInfo, OscLMemAllocator > &iOutAudFormatCapability, OscL_Vector< FormatCapabilityInfo, OscLMemAllocator > &iInVidFormatCapability, OscL_Vector< FormatCapabilityInfo, OscLMemAllocator > &iOutVidFormatCapability)=0`

### 4.11.1 Member Function Documentation

#### 4.11.1.1 virtual void PV2WayTestExtensionInterface::addRef () [pure virtual]

Increment reference counter for this interface.

#### 4.11.1.2 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]

#### 4.11.1.3 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.

#### 4.11.1.4 virtual void PV2WayTestExtensionInterface::removeRef () [pure virtual]

Decrement reference counter for this interface.

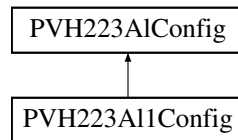
The documentation for this class was generated from the following file:

- [pv\\_2way\\_test\\_extension\\_interface.h](#)

## 4.12 PVH223A11Config Class Reference

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

Inheritance diagram for PVH223A11Config::



### Public Methods

- [PVH223A1Index](#) `IsA () const`

### Data Fields

- bool `iFramed`

### 4.12.1 Detailed Description

PVH223A11Config class

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

### 4.12.2 Member Function Documentation

#### 4.12.2.1 [PVH223A1Index](#) `PVH223A11Config::IsA () const` [inline, virtual]

Implements [PVH223A1Config](#).

### 4.12.3 Field Documentation

#### 4.12.3.1 bool `PVH223A11Config::iFramed`

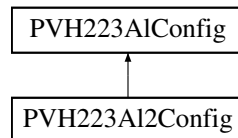
The documentation for this class was generated from the following file:

- [pv\\_2way\\_h324m\\_types.h](#)

## 4.13 PVH223A12Config Class Reference

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

Inheritance diagram for PVH223A12Config::



### Public Methods

- [PVH223A1Index](#) `IsA () const`

### Data Fields

- `bool` [iUseSequenceNumbers](#)

### 4.13.1 Detailed Description

PVH223A12Config class

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

### 4.13.2 Member Function Documentation

#### 4.13.2.1 [PVH223A1Index](#) `PVH223A12Config::IsA () const` [inline, virtual]

Implements [PVH223A1Config](#).

### 4.13.3 Field Documentation

#### 4.13.3.1 `bool` `PVH223A12Config::iUseSequenceNumbers`

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

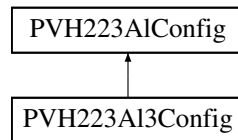
- [pv\\_2way\\_h324m\\_types.h](#)



## 4.14 PVH223A13Config Class Reference

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

Inheritance diagram for PVH223A13Config::



### Public Methods

- [PVH223A1Index](#) `IsA () const`

### Data Fields

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

### 4.14.1 Detailed Description

PVH223A13Config class

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

### 4.14.2 Member Function Documentation

#### 4.14.2.1 [PVH223A1Index](#) `PVH223A13Config::IsA () const` [inline, virtual]

Implements [PVH223A1Config](#).

### 4.14.3 Field Documentation

#### 4.14.3.1 `uint32` `PVH223A13Config::iControlFieldOctets`

#### 4.14.3.2 `uint32` `PVH223A13Config::iSendBufferSize`

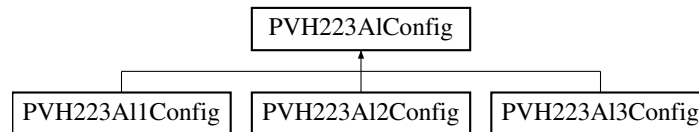
The documentation for this class was generated from the following file:

- [pv\\_2way\\_h324m\\_types.h](#)

## 4.15 PVH223AIConfig Class Reference

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

Inheritance diagram for PVH223AIConfig::



### Public Types

- enum [PVH223AIIndex](#) { [PVH223\\_AL1](#) = 1, [PVH223\\_AL2](#) = 2, [PVH223\\_AL3](#) = 4 }

### Public Methods

- virtual [PVH223AIIndex IsA](#) () const=0

### 4.15.1 Detailed Description

PVH223AIConfig class

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

### 4.15.2 Member Enumeration Documentation

#### 4.15.2.1 enum PVH223AIConfig::PVH223AIIndex

Enumeration values:

[PVH223\\_AL1](#)

[PVH223\\_AL2](#)

[PVH223\\_AL3](#)

### 4.15.3 Member Function Documentation

#### 4.15.3.1 virtual [PVH223AIIndex](#) PVH223AIConfig::IsA () [pure virtual]

Implemented in [PVH223AI1Config](#), [PVH223AI2Config](#), and [PVH223AI3Config](#).

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

- [pv\\_2way\\_h324m\\_types.h](#)

## 4.16 PVMFComponentInterface Class Reference

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

### Public Methods

- [PVMFComponentInterface \(\)](#)
- void [addRef \(\)](#)
- void [removeRef \(\)](#)

### Protected Attributes

- int32 [iReferenceCount](#)

#### 4.16.1 Constructor & Destructor Documentation

4.16.1.1 [PVMFComponentInterface::PVMFComponentInterface \(\)](#) [inline]

#### 4.16.2 Member Function Documentation

4.16.2.1 [void PVMFComponentInterface::addRef \(\)](#) [inline]

4.16.2.2 [void PVMFComponentInterface::removeRef \(\)](#) [inline]

#### 4.16.3 Field Documentation

4.16.3.1 [int32 PVMFComponentInterface::iReferenceCount](#) [protected]

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

- [tsc\\_h324m\\_config\\_interface.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"
```

#### 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\_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](#) }

#### 5.3.1 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 enumerates the mode the peer wants to transition to after the disconnect

### 5.3.2.2 typedef enum **TPVUserInputType** PV2WayUserInputType

TPVUserInputType enum Enumeration of user input types

## 5.3.3 Enumeration Type Documentation

### 5.3.3.1 enum TPVPostDisconnectOption

TPVPostDisconnectOption Enum

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

**Enumeration values:**

**EDisconnectLine**

**EAnalogueTelephony**

### 5.3.3.2 enum TPVUserInputType

TPVUserInputType enum Enumeration of user input types

**Enumeration values:**

**EAlphanumeric**

**EDtmf**

## 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 "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,0x86,0x28,0xa7,0x69,0x92,0xe3)

### 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"
```

### Data Structures

- class [H324MConfigInterface](#)
- class [H324MConfigObserver](#)
- class [H324MReverseParametersExtensionInterface](#)
- class [PVMFComponentInterface](#)

### 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,0x98,0x05,0x3f)

### 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](#) }

## 5.7.1 Define Documentation

- 5.7.1.1 **#define PVH324MConfigUuid** PVU-uid(0x2b0b54e2,0x7079,0x46c6,0xb2,0x3e,0x04,0xff,0xd3,0x0e,0x14,0x36)
- 5.7.1.2 **#define PVUuidH324ComponentInterface** PVU-uid(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.

#### 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.



# Index

- ~CPV2WayInterface
  - CPV2WayInterface, 7
- ~H324MConfigObserver
  - H324MConfigObserver, 25
- ~PV2Way324ConnectOptions
  - PV2Way324ConnectOptions, 28
- ~PV2Way324InitInfo
  - PV2Way324InitInfo, 29
- ~PV2WayInitInfo
  - PV2WayInitInfo, 33
- AddDataSink
  - CPV2WayInterface, 7
- AddDataSource
  - CPV2WayInterface, 7
- addRef
  - PV2WayTestExtensionInterface, 35
  - PVMFComponentInterface, 40
- 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, 48
- EAnalogueTelephony
  - pv\_2way\_h324m\_types.h, 48
- EConnected
  - pv\_2way\_basic\_types.h, 43
- EConnecting
  - pv\_2way\_basic\_types.h, 43
- EDisconnecting
  - pv\_2way\_basic\_types.h, 43
- EDisconnectLine
  - pv\_2way\_h324m\_types.h, 48
- EDtmf
  - pv\_2way\_h324m\_types.h, 48
- EIdle
  - pv\_2way\_basic\_types.h, 42
- EInitializing
  - pv\_2way\_basic\_types.h, 42
- EResetting

- pv\_2way\_basic\_types.h, 43
- ESetup
  - pv\_2way\_basic\_types.h, 43
- FastUpdate
  - H324MConfigInterface, 17
- GetConnectInfoClassName
  - PV2Way324ConnectOptions, 28
  - PV2WayConnectOptions, 32
- GetFormatCapabilities
  - H324MReverseParametersExtensionInterface, 26
- GetInitInfoClassName
  - PV2Way324InitInfo, 29
  - PV2WayInitInfo, 33
- GetLogLevel
  - CPV2WayInterface, 9
- GetPortTag
  - H324MReverseParametersExtensionInterface, 26
- GetSDKInfo
  - CPV2WayInterface, 9
- GetSDKModuleInfo
  - CPV2WayInterface, 9
- GetState
  - CPV2WayInterface, 10
- H324MConfigCommandCompletedL
  - H324MConfigObserver, 25
- H324MConfigHandleInformationalEventL
  - H324MConfigObserver, 25
- H324MConfigInformationalEvent
  - tsc\_h324m\_config\_interface.h, 53
- H324MConfigInterface, 16
- H324MConfigInterface
  - FastUpdate, 17
  - SendEndSession, 17
  - SendRme, 17
  - SendRtd, 17
  - SendSkewIndication, 18
  - SendUserInput, 18
  - SendVendorId, 18
  - SendVideoTemporalSpatialTradeoffCommand, 18
  - SendVideoTemporalSpatialTradeoffIndication, 18
  - SetA12SequenceNumbers, 19
  - SetA13ControlFieldOctets, 19
  - SetALConfiguration, 19
  - SetCodecPreference, 19
  - SetEndSessionTimeout, 20
  - SetLogicalChannelBufferingMs, 20
  - SetMaxMuxCcsrcsduSize, 20
  - SetMaxMuxPduSize, 21
  - SetMaxPduSize, 21
  - SetMaxSduSize, 21
  - SetMaxSduSizeR, 21
  - SetMultiplexLevel, 22
  - SetObserver, 22
  - SetTerminalType, 22
  - SetTimerCounter, 22
  - SetVendor, 23
  - SetVideoResolutions, 23
  - SetWnsrp, 23
- H324MConfigObserver, 25
- H324MConfigObserver
  - ~H324MConfigObserver, 25
- H324MConfigCommandCompletedL, 25
- H324MConfigHandleInformationalEventL, 25
- H324MConfigStatusResponse
  - tsc\_h324m\_config\_interface.h, 53
- H324MReverseParametersExtensionInterface, 26
- H324MReverseParametersExtensionInterface
  - GetFormatCapabilities, 26
  - GetPortTag, 26
  - SetPortTag, 26
- iControlFieldOctets
  - PVH223A13Config, 38
- iDisconnectTimeoutInterval
  - PV2Way324ConnectOptions, 28
- iFramed
  - PVH223A11Config, 36
- iIncomingAudioFormats
  - PV2WayInitInfo, 33
- iIncomingVideoFormats
  - PV2WayInitInfo, 33
- iLocalId
  - PV2WayConnectOptions, 32
- iLocalIdSize
  - PV2WayConnectOptions, 32
- iLoopbackMode
  - PV2WayConnectOptions, 32
- iMultiplexingDelayMs
  - PV2Way324InitInfo, 30
- INCOMING
  - pv\_2way\_basic\_types.h, 43
- Init
  - CPV2WayEngineFactory, 5
  - CPV2WayInterface, 10
  - CPV2WayProxyFactory, 15
- iOutgoingAudioFormats
  - PV2WayInitInfo, 34
- iOutgoingVideoFormats
  - PV2WayInitInfo, 34

- iReferenceCount
  - PVMFCComponentInterface, 40
- iRemoteId
  - PV2WayConnectOptions, 32
- iRemoteIdSize
  - PV2WayConnectOptions, 32
- IsA
  - PVH223A11Config, 36
  - PVH223A12Config, 37
  - PVH223A13Config, 38
  - PVH223A1Config, 39
- iSendBufferSize
  - PVH223A13Config, 38
- iUseSequenceNumbers
  - PVH223A12Config, 37
- NegotiatedFormatsMatch
  - PV2WayTestExtensionInterface, 35
- OUTGOING
  - pv\_2way\_basic\_types.h, 43
- Pause
  - CPV2WayInterface, 10
- PV2Way324ConnectOptions, 27
  - PV2Way324ConnectOptions, 27
- PV2Way324ConnectOptions
  - ~PV2Way324ConnectOptions, 28
  - GetConnectInfoClassName, 28
  - iDisconnectTimeoutInterval, 28
  - PV2Way324ConnectOptions, 27
- PV2Way324InitInfo, 29
  - PV2Way324InitInfo, 29
- PV2Way324InitInfo
  - ~PV2Way324InitInfo, 29
  - GetInitInfoClassName, 29
  - iMultiplexingDelayMs, 30
  - PV2Way324InitInfo, 29
- PV2WayConnectOptions, 31
  - PV2WayConnectOptions, 31
- PV2WayConnectOptions
  - GetConnectInfoClassName, 32
  - iLocalId, 32
  - iLocalIdSize, 32
  - iLoopbackMode, 32
  - iRemoteId, 32
  - iRemoteIdSize, 32
  - PV2WayConnectOptions, 31
- PV2WayDirection
  - pv\_2way\_basic\_types.h, 42
- PV2WayDispatchError
  - pv\_2way\_basic\_types.h, 45
- PV2WayErrorRejected
  - pv\_2way\_basic\_types.h, 45
- PV2WayErrorStatusStart
  - pv\_2way\_basic\_types.h, 45
- PV2WayErrReplaced
  - pv\_2way\_basic\_types.h, 45
- PV2WayInitInfo, 33
- PV2WayInitInfo
  - ~PV2WayInitInfo, 33
  - GetInitInfoClassName, 33
  - iIncomingAudioFormats, 33
  - iIncomingVideoFormats, 33
  - iOutgoingAudioFormats, 34
  - iOutgoingVideoFormats, 34
- PV2WayLoopbackMode
  - pv\_2way\_basic\_types.h, 42
- PV2WayMediaType
  - pv\_2way\_basic\_types.h, 42
- PV2WayPostDisconnectOption
  - pv\_2way\_h324m\_types.h, 47
- PV2WayState
  - pv\_2way\_basic\_types.h, 42
- PV2WayTerminalType
  - pv\_2way\_basic\_types.h, 42
- PV2WayTestEncExtensionUUID
  - pv\_2way\_test\_extension\_interface.h, 51
- PV2WayTestExtensionInterface, 35
- PV2WayTestExtensionInterface
  - addRef, 35
  - NegotiatedFormatsMatch, 35
  - queryInterface, 35
  - removeRef, 35
- PV2WayUserInputType
  - pv\_2way\_h324m\_types.h, 47
- pv\_2way\_basic\_types.h
  - EConnected, 43
  - EConnecting, 43
  - EDisconnecting, 43
  - EIdle, 42
  - EInitializing, 42
  - EResetting, 43
  - ESetup, 43
  - INCOMING, 43
  - OUTGOING, 43
  - PV\_323, 44
  - PV\_324M, 44
  - PV\_AUDIO, 44
  - PV\_CONTROL, 44
  - PV\_DATA, 44
  - PV\_DIRECTION\_BOTH, 43
  - PV\_DIRECTION\_NONE, 43
  - PV\_LOOPBACK\_COMM, 43
  - PV\_LOOPBACK\_ENGINE, 43
  - PV\_LOOPBACK\_MUX, 43
  - PV\_LOOPBACK\_NONE, 43
  - PV\_MEDIA\_ALL, 44

- PV\_MEDIA\_NONE, 44
- PV\_MULTIPLEXED, 44
- PV\_SIP, 44
- PV\_TERMINAL\_TYPE\_NONE, 44
- PV\_USER\_INPUT, 44
- PV\_VIDEO, 44
- PVT\_INDICATION\_CLOSE\_TRACK, 44
- PVT\_INDICATION\_CLOSING\_TRACK, 44
- PVT\_INDICATION\_DISCONNECT, 44
- PVT\_INDICATION\_INCOMING\_TRACK, 44
- PVT\_INDICATION\_INTERNAL\_ERROR, 45
- PVT\_INDICATION\_OUTGOING\_TRACK, 44
- PVT\_INDICATION\_PAUSE\_TRACK, 44
- PVT\_INDICATION\_RESUME\_TRACK, 45
- pv\_2way\_basic\_types.h, 41
  - PV2WayDirection, 42
  - PV2WayDispatchError, 45
  - PV2WayErrorRejected, 45
  - PV2WayErrorStatusStart, 45
  - PV2WayErrReplaced, 45
  - PV2WayLoopbackMode, 42
  - PV2WayMediaType, 42
  - PV2WayState, 42
  - PV2WayTerminalType, 42
  - PVTrackId, 42
  - TPVDirection, 43
  - TPVLoopbackMode, 43
  - TPVMediaType\_t, 43
  - TPVTerminalType, 44
  - TPVTIndicationType, 44
- pv\_2way\_engine\_factory.h, 46
- pv\_2way\_h324m\_types.h
  - EAlphanumeric, 48
  - EAnalogueTelephony, 48
  - EDisconnectLine, 48
  - EDtmf, 48
- pv\_2way\_h324m\_types.h, 47
  - PV2WayPostDisconnectOption, 47
  - PV2WayUserInputType, 47
  - PV\_2WAY\_MAX\_SKEW\_MS, 47
  - PV\_2WAY\_MAX\_USER\_INPUT\_FORMATS, 47
  - TPVPostDisconnectOption, 48
  - TPVUserInputType, 48
- pv\_2way\_interface.h, 49
- PV\_2WAY\_MAX\_SKEW\_MS
  - pv\_2way\_h324m\_types.h, 47
- PV\_2WAY\_MAX\_USER\_INPUT\_FORMATS
  - pv\_2way\_h324m\_types.h, 47
- pv\_2way\_proxy\_factory.h, 50
- pv\_2way\_test\_extension\_interface.h, 51
  - PV2WayTestEncExtensionUUID, 51
- PV\_323
  - pv\_2way\_basic\_types.h, 44
- PV\_324M
  - pv\_2way\_basic\_types.h, 44
- PV\_AUDIO
  - pv\_2way\_basic\_types.h, 44
- PV\_CONTROL
  - pv\_2way\_basic\_types.h, 44
- PV\_DATA
  - pv\_2way\_basic\_types.h, 44
- PV\_DIRECTION\_BOTH
  - pv\_2way\_basic\_types.h, 43
- PV\_DIRECTION\_NONE
  - pv\_2way\_basic\_types.h, 43
- PV\_INDICATION\_FAST\_UPDATE
  - tsc\_h324m\_config\_interface.h, 53
- PV\_INDICATION\_RME
  - tsc\_h324m\_config\_interface.h, 53
- PV\_INDICATION\_RTD
  - tsc\_h324m\_config\_interface.h, 53
- PV\_INDICATION\_SKEW
  - tsc\_h324m\_config\_interface.h, 53
- PV\_INDICATION\_USER\_INPUT
  - tsc\_h324m\_config\_interface.h, 53
- PV\_INDICATION\_USER\_INPUT\_CAPABILITY
  - tsc\_h324m\_config\_interface.h, 53
- PV\_INDICATION\_VENDOR\_ID
  - tsc\_h324m\_config\_interface.h, 53
- PV\_INDICATION\_VIDEO\_SPATIAL\_TEMPORAL\_TRADEOFF\_COMMAND
  - tsc\_h324m\_config\_interface.h, 53
- PV\_INDICATION\_VIDEO\_SPATIAL\_TEMPORAL\_TRADEOFF\_INDICATION
  - tsc\_h324m\_config\_interface.h, 53
- PV\_LOOPBACK\_COMM
  - pv\_2way\_basic\_types.h, 43
- PV\_LOOPBACK\_ENGINE
  - pv\_2way\_basic\_types.h, 43
- PV\_LOOPBACK\_MUX
  - pv\_2way\_basic\_types.h, 43
- PV\_LOOPBACK\_NONE
  - pv\_2way\_basic\_types.h, 43
- PV\_MEDIA\_ALL
  - pv\_2way\_basic\_types.h, 44
- PV\_MEDIA\_NONE
  - pv\_2way\_basic\_types.h, 44
- PV\_MULTIPLEXED
  - pv\_2way\_basic\_types.h, 44

- PV\_SIP
  - pv\_2way\_basic\_types.h, 44
- PV\_TERMINAL\_TYPE\_NONE
  - pv\_2way\_basic\_types.h, 44
- PV\_USER\_INPUT
  - pv\_2way\_basic\_types.h, 44
- PV\_VIDEO
  - pv\_2way\_basic\_types.h, 44
- PVH223\_AL1
  - PVH223AlConfig, 39
- PVH223\_AL2
  - PVH223AlConfig, 39
- PVH223\_AL3
  - PVH223AlConfig, 39
- PVH223Al1Config, 36
  - iFramed, 36
  - IsA, 36
- PVH223Al2Config, 37
  - IsA, 37
  - iUseSequenceNumbers, 37
- PVH223Al3Config, 38
  - iControlFieldOctets, 38
  - IsA, 38
  - iSendBufferSize, 38
- PVH223AlConfig, 39
  - PVH223\_AL1, 39
  - PVH223\_AL2, 39
  - PVH223\_AL3, 39
- PVH223AlConfig
  - IsA, 39
  - PVH223AlIndex, 39
- PVH223AlIndex
  - PVH223AlConfig, 39
- PVH324MConfigUuid
  - tsc\_h324m\_config\_interface.h, 53
- PVH324MIndicationType
  - tsc\_h324m\_config\_interface.h, 53
- PVMFComponentInterface, 40
  - PVMFComponentInterface, 40
- PVMFComponentInterface
  - addRef, 40
  - iReferenceCount, 40
  - PVMFComponentInterface, 40
  - removeRef, 40
- PVT\_INDICATION\_CLOSE\_TRACK
  - pv\_2way\_basic\_types.h, 44
- PVT\_INDICATION\_CLOSING\_TRACK
  - pv\_2way\_basic\_types.h, 44
- PVT\_INDICATION\_DISCONNECT
  - pv\_2way\_basic\_types.h, 44
- PVT\_INDICATION\_INCOMING\_TRACK
  - pv\_2way\_basic\_types.h, 44
- PVT\_INDICATION\_INTERNAL\_ERROR
  - pv\_2way\_basic\_types.h, 45
- PVT\_INDICATION\_OUTGOING\_TRACK
  - pv\_2way\_basic\_types.h, 44
- PVT\_INDICATION\_PAUSE\_TRACK
  - pv\_2way\_basic\_types.h, 44
- PVT\_INDICATION\_RESUME\_TRACK
  - pv\_2way\_basic\_types.h, 45
- PVTrackId
  - pv\_2way\_basic\_types.h, 42
- PVUuidH324ComponentInterface
  - tsc\_h324m\_config\_interface.h, 53
- QueryInterface
  - CPV2WayInterface, 11
- queryInterface
  - PV2WayTestExtensionInterface, 35
- QueryUUID
  - CPV2WayInterface, 11
- RemoveDataSink
  - CPV2WayInterface, 11
- RemoveDataSource
  - CPV2WayInterface, 12
- RemoveLogAppender
  - CPV2WayInterface, 12
- removeRef
  - PV2WayTestExtensionInterface, 35
  - PVMFComponentInterface, 40
- 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
- SetAl2SequenceNumbers
  - H324MConfigInterface, 19
- SetAl3ControlFieldOctets
  - H324MConfigInterface, 19
- SetAlConfiguration
  - H324MConfigInterface, 19

- SetCodecPreference
  - H324MConfigInterface, [19](#)
- SetEndSessionTimeout
  - H324MConfigInterface, [20](#)
- SetLogAppender
  - CPV2WayInterface, [13](#)
- SetLogicalChannelBufferingMs
  - H324MConfigInterface, [20](#)
- SetLogLevel
  - CPV2WayInterface, [13](#)
- SetMaxMuxCcsrlSduSize
  - H324MConfigInterface, [20](#)
- SetMaxMuxPduSize
  - H324MConfigInterface, [21](#)
- SetMaxPduSize
  - H324MConfigInterface, [21](#)
- SetMaxSduSize
  - H324MConfigInterface, [21](#)
- SetMaxSduSizeR
  - H324MConfigInterface, [21](#)
- SetMultiplexLevel
  - H324MConfigInterface, [22](#)
- SetObserver
  - H324MConfigInterface, [22](#)
- SetPortTag
  - H324MReverseParametersExtension-Interface, [26](#)
- SetTerminalType
  - H324MConfigInterface, [22](#)
- SetTimerCounter
  - H324MConfigInterface, [22](#)
- SetVendor
  - H324MConfigInterface, [23](#)
- SetVideoResolutions
  - H324MConfigInterface, [23](#)
- SetWnsrp
  - H324MConfigInterface, [23](#)
  
- TPVDirection
  - pv\_2way\_basic\_types.h, [43](#)
- TPVLoopbackMode
  - pv\_2way\_basic\_types.h, [43](#)
- TPVMediaType\_t
  - pv\_2way\_basic\_types.h, [43](#)
- TPVPostDisconnectOption
  - pv\_2way\_h324m\_types.h, [48](#)
- TPVTerminalType
  - pv\_2way\_basic\_types.h, [44](#)
- TPVTIndicationType
  - pv\_2way\_basic\_types.h, [44](#)
- TPVUserInputType
  - pv\_2way\_h324m\_types.h, [48](#)
- tsc\_h324m\_config\_interface.h
  - PV\_INDICATION\_FAST\_UPDATE, [53](#)
  - PV\_INDICATION\_RME, [53](#)
  - PV\_INDICATION\_RTD, [53](#)
  - PV\_INDICATION\_SKEW, [53](#)
  - PV\_INDICATION\_USER\_INPUT, [53](#)
  - PV\_INDICATION\_USER\_INPUT\_-CAPABILITY, [53](#)
  - PV\_INDICATION\_VENDOR\_ID, [53](#)
  - PV\_INDICATION\_VIDEO\_SPATIAL\_-TEMPORAL\_TRADEOFF\_-COMMAND, [53](#)
  - PV\_INDICATION\_VIDEO\_SPATIAL\_-TEMPORAL\_TRADEOFF\_-INDICATION, [53](#)
- tsc\_h324m\_config\_interface.h, [52](#)
  - H324MConfigInformationalEvent, [53](#)
  - H324MConfigStatusResponse, [53](#)
  - PVH324MConfigUuid, [53](#)
  - PVH324MIndicationType, [53](#)
  - PVUuidH324ComponentInterface, [53](#)