



Open CORE™

PV Author Engine

Build Version: CORE_9.000.1.1_RC3

May 14, 2010

Contents

1	Data Structure Index	1
1.1	Class Hierarchy	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	7
4.1	CPVCmnAsyncEvent Class Reference	7
4.1.1	Detailed Description	7
4.1.2	Constructor & Destructor Documentation	8
4.1.2.1	CPVCmnAsyncEvent	8
4.1.2.2	~CPVCmnAsyncEvent	8
4.1.3	Member Function Documentation	8
4.1.3.1	GetEventData	8
4.1.3.2	GetEventType	8
4.1.3.3	GetLocalBuffer	8
4.1.4	Field Documentation	8
4.1.4.1	iEventType	8
4.1.4.2	iExclusivePtr	8
4.1.4.3	iLocalBuffer	8
4.2	CPVCmnCmdResp Class Reference	10
4.2.1	Constructor & Destructor Documentation	10
4.2.1.1	CPVCmnCmdResp	10
4.2.2	Member Function Documentation	10
4.2.2.1	GetCmdId	10
4.2.2.2	GetCmdStatus	11

4.2.2.3	GetCmdType	11
4.2.2.4	GetContext	11
4.2.2.5	GetResponseData	11
4.2.2.6	GetResponseDataSize	11
4.2.3	Field Documentation	11
4.2.3.1	iCmdId	11
4.2.3.2	iCmdType	11
4.2.3.3	iContext	11
4.2.3.4	iResponseData	12
4.2.3.5	iResponseDataSize	12
4.2.3.6	iStatus	12
4.3	CPVCmnInterfaceCmdMessage Class Reference	13
4.3.1	Detailed Description	13
4.3.2	Constructor & Destructor Documentation	13
4.3.2.1	CPVCmnInterfaceCmdMessage	13
4.3.2.2	CPVCmnInterfaceCmdMessage	13
4.3.2.3	~CPVCmnInterfaceCmdMessage	14
4.3.3	Member Function Documentation	14
4.3.3.1	compare	14
4.3.3.2	GetCommandId	14
4.3.3.3	GetContextData	14
4.3.3.4	GetPriority	14
4.3.3.5	GetType	14
4.3.3.6	SetId	14
4.3.4	Friends And Related Function Documentation	14
4.3.4.1	operator<	14
4.3.4.2	PVInterfaceProxy	14
4.3.5	Field Documentation	14
4.3.5.1	iContextData	14
4.3.5.2	iId	14
4.3.5.3	iPriority	14
4.3.5.4	iType	15
4.4	CPVCmnInterfaceObserverMessage Class Reference	16
4.4.1	Detailed Description	16
4.4.2	Constructor & Destructor Documentation	17
4.4.2.1	CPVCmnInterfaceObserverMessage	17

4.4.2.2	CPVCmnInterfaceObserverMessage	17
4.4.2.3	~CPVCmnInterfaceObserverMessage	17
4.4.3	Member Function Documentation	17
4.4.3.1	GetPriority	17
4.4.3.2	GetResponseType	17
4.4.4	Field Documentation	17
4.4.4.1	iOrder	17
4.4.4.2	iPriority	17
4.4.4.3	iResponseType	17
4.5	CPVCmnInterfaceObserverMessageCompare Class Reference	18
4.5.1	Member Function Documentation	18
4.5.1.1	compare	18
4.6	MPVCmnCmdStatusObserver Class Reference	19
4.6.1	Constructor & Destructor Documentation	19
4.6.1.1	~MPVCmnCmdStatusObserver	19
4.6.2	Member Function Documentation	19
4.6.2.1	CommandCompletedL	19
4.7	MPVCmnErrorEventObserver Class Reference	20
4.7.1	Constructor & Destructor Documentation	20
4.7.1.1	~MPVCmnErrorEventObserver	20
4.7.2	Member Function Documentation	20
4.7.2.1	HandleErrorEventL	20
4.8	MPVCmnInfoEventObserver Class Reference	21
4.8.1	Constructor & Destructor Documentation	21
4.8.1.1	~MPVCmnInfoEventObserver	21
4.8.2	Member Function Documentation	21
4.8.2.1	HandleInformationalEventL	21
4.9	PVAsyncErrorEvent Class Reference	22
4.9.1	Detailed Description	22
4.9.2	Constructor & Destructor Documentation	22
4.9.2.1	PVAsyncErrorEvent	22
4.9.2.2	PVAsyncErrorEvent	22
4.9.2.3	~PVAsyncErrorEvent	22
4.9.3	Member Function Documentation	22
4.9.3.1	GetEventData	22
4.9.3.2	GetEventType	23

4.9.3.3	GetResponseType	23
4.10	PVAsyncInformationalEvent Class Reference	24
4.10.1	Detailed Description	24
4.10.2	Constructor & Destructor Documentation	24
4.10.2.1	PVAsyncInformationalEvent	24
4.10.2.2	PVAsyncInformationalEvent	24
4.10.2.3	~PVAsyncInformationalEvent	24
4.10.3	Member Function Documentation	24
4.10.3.1	GetEventData	24
4.10.3.2	GetEventType	25
4.10.3.3	GetResponseType	25
4.11	PVAuthorEngineFactory Class Reference	26
4.11.1	Detailed Description	26
4.11.2	Member Function Documentation	26
4.11.2.1	CreateAuthor	26
4.11.2.2	DeleteAuthor	26
4.12	PVAuthorEngineInterface Class Reference	28
4.12.1	Detailed Description	29
4.12.2	Constructor & Destructor Documentation	29
4.12.2.1	~PVAuthorEngineInterface	29
4.12.3	Member Function Documentation	29
4.12.3.1	AddDataSink	29
4.12.3.2	AddDataSource	29
4.12.3.3	AddMediaTrack	30
4.12.3.4	AddMediaTrack	30
4.12.3.5	CancelAllCommands	31
4.12.3.6	Close	31
4.12.3.7	GetLogLevel	31
4.12.3.8	GetPVAuthorState	32
4.12.3.9	GetSDKInfo	32
4.12.3.10	GetSDKModuleInfo	32
4.12.3.11	Init	33
4.12.3.12	Open	33
4.12.3.13	Pause	33
4.12.3.14	QueryInterface	34
4.12.3.15	RemoveDataSink	34

4.12.3.16 RemoveDataSource	34
4.12.3.17 RemoveLogAppender	35
4.12.3.18 Reset	35
4.12.3.19 Resume	36
4.12.3.20 SelectComposer	36
4.12.3.21 SelectComposer	36
4.12.3.22 SetLogAppender	37
4.12.3.23 SetLogLevel	37
4.12.3.24 Start	38
4.12.3.25 Stop	38
4.13 PVCmdResponse Class Reference	39
4.13.1 Detailed Description	39
4.13.2 Constructor & Destructor Documentation	39
4.13.2.1 PVCmdResponse	39
4.13.2.2 PVCmdResponse	39
4.13.3 Member Function Documentation	39
4.13.3.1 GetCmdId	39
4.13.3.2 GetCmdStatus	40
4.13.3.3 GetContext	40
4.13.3.4 GetExtendedErrorInfoMessage	40
4.13.3.5 GetResponseData	40
4.13.3.6 GetResponseDataSize	40
4.13.3.7 GetResponseType	40
4.14 PVCommandStatusObserver Class Reference	41
4.14.1 Detailed Description	41
4.14.2 Constructor & Destructor Documentation	41
4.14.2.1 ~PVCommandStatusObserver	41
4.14.3 Member Function Documentation	41
4.14.3.1 CommandCompleted	41
4.15 PVConfigInterface Class Reference	42
4.15.1 Detailed Description	42
4.16 PVEngineAsyncEvent Class Reference	43
4.16.1 Detailed Description	43
4.16.2 Constructor & Destructor Documentation	43
4.16.2.1 PVEngineAsyncEvent	43
4.16.2.2 PVEngineAsyncEvent	43

4.16.3	Member Function Documentation	44
4.16.3.1	GetAsyncEventType	44
4.16.4	Field Documentation	44
4.16.4.1	iAsyncEventType	44
4.17	PVEngineCommand Class Reference	45
4.17.1	Detailed Description	45
4.17.2	Constructor & Destructor Documentation	45
4.17.2.1	PVEngineCommand	45
4.17.2.2	PVEngineCommand	46
4.17.3	Member Function Documentation	46
4.17.3.1	GetCmdId	46
4.17.3.2	GetCmdType	46
4.17.3.3	GetContext	46
4.17.3.4	GetMimeType	47
4.17.3.5	GetParam1	47
4.17.3.6	GetParam2	47
4.17.3.7	GetParam3	47
4.17.3.8	GetUuid	47
4.17.3.9	SetMimeType	48
4.17.3.10	SetUuid	48
4.17.4	Field Documentation	48
4.17.4.1	iCmdId	48
4.17.4.2	iCmdType	48
4.17.4.3	iContextData	48
4.17.4.4	iMimeType	48
4.17.4.5	iParam1	48
4.17.4.6	iParam2	48
4.17.4.7	iParam3	48
4.17.4.8	iUuid	48
4.18	PVErrorEventObserver Class Reference	50
4.18.1	Detailed Description	50
4.18.2	Constructor & Destructor Documentation	50
4.18.2.1	~PVErrorEventObserver	50
4.18.3	Member Function Documentation	50
4.18.3.1	HandleErrorEvent	50
4.19	PVInformationalEventObserver Class Reference	51

4.19.1	Detailed Description	51
4.19.2	Constructor & Destructor Documentation	51
4.19.2.1	~PVInformationalEventObserver	51
4.19.3	Member Function Documentation	51
4.19.3.1	HandleInformationalEvent	51
4.20	PVSDKInfo Struct Reference	52
4.20.1	Constructor & Destructor Documentation	52
4.20.1.1	PVSDKInfo	52
4.20.2	Member Function Documentation	52
4.20.2.1	operator=	52
4.20.3	Field Documentation	52
4.20.3.1	iDate	52
4.20.3.2	iLabel	52
4.21	TPVCmnSDKInfo Struct Reference	53
4.21.1	Constructor & Destructor Documentation	53
4.21.1.1	TPVCmnSDKInfo	53
4.21.2	Member Function Documentation	53
4.21.2.1	operator=	53
4.21.3	Field Documentation	53
4.21.3.1	iDate	53
4.21.3.2	iLabel	53
5	File Documentation	55
5.1	pv_common_types.h File Reference	55
5.1.1	Define Documentation	56
5.1.1.1	PV_COMMON_ASYNC_EVENT_LOCAL_BUF_SIZE	56
5.1.2	Typedef Documentation	56
5.1.2.1	CPVCmnAsyncErrorEvent	56
5.1.2.2	CPVCmnAsyncInfoEvent	56
5.1.2.3	CPVCmnAudioCaps	56
5.1.2.4	CPVCmnAudioPrefs	56
5.1.2.5	CPVCmnVideoCaps	56
5.1.2.6	CPVCmnVideoPrefs	56
5.1.2.7	TPVCmnCommandId	56
5.1.2.8	TPVCmnCommandStatus	56
5.1.2.9	TPVCmnCommandType	56
5.1.2.10	TPVCmnEventType	56

5.1.2.11	TPVCmnExclusivePtr	56
5.1.2.12	TPVCmnInterfacePtr	56
5.1.2.13	TPVCmnMIMEType	56
5.1.2.14	TPVCmnResponseType	56
5.1.2.15	TPVCmnSDKModuleInfo	56
5.1.2.16	TPVCmnUUID	56
5.2	pv_config_interface.h File Reference	57
5.3	pv_engine_observer.h File Reference	58
5.4	pv_engine_observer_message.h File Reference	59
5.5	pv_engine_types.h File Reference	60
5.5.1	Typedef Documentation	60
5.5.1.1	PVCommandId	60
5.5.1.2	PVEventType	60
5.5.1.3	PVExclusivePtr	60
5.5.1.4	PVLogLevelInfo	60
5.5.1.5	PVPMetadataList	60
5.5.1.6	PVResponseType	60
5.5.1.7	PVSDKModuleInfo	60
5.6	pv_interface_cmd_message.h File Reference	61
5.6.1	Function Documentation	61
5.6.1.1	operator<	61
5.7	pvauthenginefactory.h File Reference	62
5.8	pvauthengineinterface.h File Reference	63
5.8.1	Enumeration Type Documentation	63
5.8.1.1	PVAEErrorEvent	63
5.8.1.2	PVAEInfoEvent	63
5.8.1.3	PVAEState	63

Chapter 1

Data Structure Index

1.1 Class Hierarchy

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

CPVCmnInterfaceCmdMessage	13
CPVCmnInterfaceObserverMessage	16
CPVCmnAsyncEvent	7
CPVCmnCmdResp	10
CPVCmnInterfaceObserverMessageCompare	18
MPVCmnCmdStatusObserver	19
MPVCmnErrorEventObserver	20
MPVCmnInfoEventObserver	21
PVAsyncErrorEvent	22
PVAsyncInformationalEvent	24
PVAuthorEngineFactory	26
PVAuthorEngineInterface	28
PVCmdResponse	39
PVCommandStatusObserver	41
PVConfigInterface	42
PVEngineAsyncEvent	43
PVEngineCommand	45
PVErrorEventObserver	50
PVInformationalEventObserver	51
PVSDKInfo	52
TPVCmnSDKInfo	53

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

CPVCmnAsyncEvent	7
CPVCmnCmdResp	10
CPVCmnInterfaceCmdMessage	13
CPVCmnInterfaceObserverMessage	16
CPVCmnInterfaceObserverMessageCompare	18
MPVCmnCmdStatusObserver	19
MPVCmnErrorEventObserver	20
MPVCmnInfoEventObserver	21
PVAsyncErrorEvent	22
PVAsyncInformationalEvent	24
PVAuthorEngineFactory	26
PVAuthorEngineInterface	28
PVCmdResponse	39
PVCommandStatusObserver	41
PVConfigInterface	42
PVEngineAsyncEvent	43
PVEngineCommand	45
PVErrorEventObserver	50
PVInformationalEventObserver	51
PVSDKInfo	52
TPVCmnSDKInfo	53

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

pv_common_types.h	55
pv_config_interface.h	57
pv_engine_observer.h	58
pv_engine_observer_message.h	59
pv_engine_types.h	60
pv_interface_cmd_message.h	61
pvauthenginefactory.h	62
pvauthengineinterface.h	63

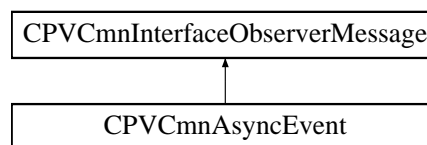
Chapter 4

Data Structure Documentation

4.1 CPVCmnAsyncEvent Class Reference

```
#include <pv_common_types.h>
```

Inheritance diagram for CPVCmnAsyncEvent:



Public Member Functions

- [CPVCmnAsyncEvent](#) ([TPVCmnEventType](#) aEventType, [TPVCmnExclusivePtr](#) aExclusivePtr, const uint8 *aLocalBuffer=NULL, uint32 aLocalBufSize=0, [TPVCmnResponseType](#) aResponseType=NULL)
- [~CPVCmnAsyncEvent](#) ()
- [TPVCmnEventType](#) [GetEventType](#) () const
- void [GetEventData](#) ([TPVCmnExclusivePtr](#) &aPtr) const
- uint8 * [GetLocalBuffer](#) ()

Protected Attributes

- [TPVCmnEventType](#) iEventType
- [TPVCmnExclusivePtr](#) iExclusivePtr
- uint8 iLocalBuffer [PV_COMMON_ASYNC_EVENT_LOCAL_BUF_SIZE]

4.1.1 Detailed Description

[CPVCmnAsyncEvent](#) Class

[CPVCmnAsyncEvent](#) is the base class used to pass unsolicited error and informational indications to the user. Additional information can be tagged based on the specific event

4.1.2 Constructor & Destructor Documentation

4.1.2.1 CPVCmnAsyncEvent::CPVCmnAsyncEvent (TPVCmnEventType *aEventType*, TPVCmnExclusivePtr *aExclusivePtr*, const uint8 * *aLocalBuffer* = NULL, uint32 *aLocalBufSize* = 0, TPVCmnResponseType *aResponseType* = NULL) [inline]

References `iLocalBuffer`, and `PV_COMMON_ASYNC_EVENT_LOCAL_BUF_SIZE`.

4.1.2.2 CPVCmnAsyncEvent::~~CPVCmnAsyncEvent () [inline]

4.1.3 Member Function Documentation

4.1.3.1 void CPVCmnAsyncEvent::GetEventData (TPVCmnExclusivePtr & *aPtr*) const [inline]

Returns

Returns the opaque data associated with the event.

References `iExclusivePtr`.

4.1.3.2 TPVCmnEventType CPVCmnAsyncEvent::GetEventType () const [inline]

Returns

Returns the Event type that has been received

References `iEventType`.

4.1.3.3 uint8* CPVCmnAsyncEvent::GetLocalBuffer () [inline]

Returns

Returns the local data associated with the event.

References `iLocalBuffer`.

4.1.4 Field Documentation

4.1.4.1 TPVCmnEventType CPVCmnAsyncEvent::iEventType [protected]

Referenced by `GetEventType()`.

4.1.4.2 TPVCmnExclusivePtr CPVCmnAsyncEvent::iExclusivePtr [protected]

Referenced by `GetEventData()`.

4.1.4.3 uint8 CPVCmnAsyncEvent::iLocalBuffer[PV_COMMON_ASYNC_EVENT_LOCAL_BUF_SIZE] [protected]

Referenced by `CPVCmnAsyncEvent()`, and `GetLocalBuffer()`.

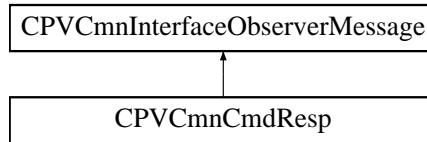
The documentation for this class was generated from the following file:

- [pv_common_types.h](#)

4.2 CPVCmnCmdResp Class Reference

```
#include <pv_common_types.h>
```

Inheritance diagram for CPVCmnCmdResp:



Public Member Functions

- [CPVCmnCmdResp](#) ([TPVCmnCommandType](#) aType, [TPVCmnCommandId](#) aId, void *aContext, [TPVCmnCommandStatus](#) aStatus, void *aResponseData=NULL, int aResponseDataSize=0, [TPVCmnResponseType](#) aResponseType=NULL)
- [TPVCmnCommandType](#) [GetCmdType](#) () const
- [TPVCmnCommandId](#) [GetCmdId](#) () const
- void * [GetContext](#) () const
- [TPVCmnCommandStatus](#) [GetCmdStatus](#) () const
- void * [GetResponseData](#) () const
- int [GetResponseDataSize](#) () const

Protected Attributes

- [TPVCmnCommandType](#) iCmdType
- [TPVCmnCommandId](#) iCmdId
- void * iContext
- [TPVCmnCommandStatus](#) iStatus
- void * iResponseData
- int iResponseDataSize

4.2.1 Constructor & Destructor Documentation

- 4.2.1.1** [CPVCmnCmdResp::CPVCmnCmdResp](#) ([TPVCmnCommandType](#) aType, [TPVCmnCommandId](#) aId, void * aContext, [TPVCmnCommandStatus](#) aStatus, void * aResponseData = NULL, int aResponseDataSize = 0, [TPVCmnResponseType](#) aResponseType = NULL) [[inline](#)]

Constructor for [CPVCmnCmdResp](#)

4.2.2 Member Function Documentation

- 4.2.2.1** [TPVCmnCommandId](#) [CPVCmnCmdResp::GetCmdId](#) () const [[inline](#)]

Returns

Returns the unique ID associated with a command of this type.

References [iCmdId](#).

4.2.2.2 TPVCmnCommandStatus CPVCmnCmdResp::GetCmdStatus () const [inline]**Returns**

Returns the completion status of the command

References iStatus.

4.2.2.3 TPVCmnCommandType CPVCmnCmdResp::GetCmdType () const [inline]**Returns**

Returns the command type that is being completed.

References iCmdType.

4.2.2.4 void* CPVCmnCmdResp::GetContext () const [inline]**Returns**

Returns the opaque data that was passed in with the command.

References iContext.

4.2.2.5 void* CPVCmnCmdResp::GetResponseData () const [inline]**Returns**

Returns additional data associated with the command. This is to be interpreted based on the command type and the return status

References iResponseData.

4.2.2.6 int CPVCmnCmdResp::GetResponseDataSize () const [inline]

References iResponseDataSize.

4.2.3 Field Documentation**4.2.3.1 TPVCmnCommandId CPVCmnCmdResp::iCmdId [protected]**

Referenced by GetCmdId().

4.2.3.2 TPVCmnCommandType CPVCmnCmdResp::iCmdType [protected]

Referenced by GetCmdType().

4.2.3.3 void* CPVCmnCmdResp::iContext [protected]

Referenced by GetContext().

4.2.3.4 void* CPVcmnCmdResp::iResponseData [protected]

Referenced by GetResponseData().

4.2.3.5 int CPVcmnCmdResp::iResponseDataSize [protected]

Referenced by GetResponseDataSize().

4.2.3.6 TPVcmnCommandStatus CPVcmnCmdResp::iStatus [protected]

Referenced by GetCmdStatus().

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

- [pv_common_types.h](#)

4.3 CPVCmnInterfaceCmdMessage Class Reference

```
#include <pv_interface_cmd_message.h>
```

Public Member Functions

- [CPVCmnInterfaceCmdMessage](#) (int aType, OsclAny *aContextData)
- [CPVCmnInterfaceCmdMessage](#) ()
- virtual [~CPVCmnInterfaceCmdMessage](#) ()
- [PVCommandId](#) GetCommandId ()
- int GetType ()
- OsclAny * GetContextData ()
- int compare (CPVCmnInterfaceCmdMessage *a, CPVCmnInterfaceCmdMessage *b) const
- int32 GetPriority () const
- void SetId (PVCommandId aId)

Protected Attributes

- [PVCommandId](#) iId
- int iType
- int32 iPriority
- OsclAny * iContextData

Friends

- class [PVInterfaceProxy](#)
- int32 operator< (const [CPVCmnInterfaceCmdMessage](#) &a, const [CPVCmnInterfaceCmdMessage](#) &b)

4.3.1 Detailed Description

CPVInterfaceCmdMessage Class

CPVInterfaceCmdMessage is the interface to the pv2way SDK, which allows initialization, control, and termination of a two-way 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

4.3.2 Constructor & Destructor Documentation

4.3.2.1 [CPVCmnInterfaceCmdMessage::CPVCmnInterfaceCmdMessage](#) (int aType, OsclAny * aContextData) [**inline**]

4.3.2.2 [CPVCmnInterfaceCmdMessage::CPVCmnInterfaceCmdMessage](#) () [**inline**]

Referenced by GetType().

4.3.2.3 `virtual CPVCmnInterfaceCmdMessage::~CPVCmnInterfaceCmdMessage () [inline, virtual]`

4.3.3 Member Function Documentation

4.3.3.1 `int CPVCmnInterfaceCmdMessage::compare (CPVCmnInterfaceCmdMessage * a, CPVCmnInterfaceCmdMessage * b) const [inline]`

The algorithm used in `OsciPriorityQueue` needs a compare function that returns true when A's priority is less than B's

Returns

true if A's priority is less than B's, else false

References `iId`.

4.3.3.2 `PVCommandId CPVCmnInterfaceCmdMessage::GetCommandId () [inline]`

4.3.3.3 `OsciAny* CPVCmnInterfaceCmdMessage::GetContextData () [inline]`

4.3.3.4 `int32 CPVCmnInterfaceCmdMessage::GetPriority () const [inline]`

References `iContextData`.

4.3.3.5 `int CPVCmnInterfaceCmdMessage::GetType () [inline]`

References `CPVCmnInterfaceCmdMessage()`.

4.3.3.6 `void CPVCmnInterfaceCmdMessage::SetId (PVCommandId aid) [inline]`

4.3.4 Friends And Related Function Documentation

4.3.4.1 `int32 operator< (const CPVCmnInterfaceCmdMessage & a, const CPVCmnInterfaceCmdMessage & b) [friend]`

4.3.4.2 `friend class PVInterfaceProxy [friend]`

4.3.5 Field Documentation

4.3.5.1 `OsciAny* CPVCmnInterfaceCmdMessage::iContextData [protected]`

Referenced by `GetPriority()`.

4.3.5.2 `PVCommandId CPVCmnInterfaceCmdMessage::iId [protected]`

Referenced by `compare()`, and `operator<()`.

4.3.5.3 `int32 CPVCmnInterfaceCmdMessage::iPriority [protected]`

Referenced by `operator<()`.

4.3.5.4 int CPVCmnInterfaceCmdMessage::iType [protected]

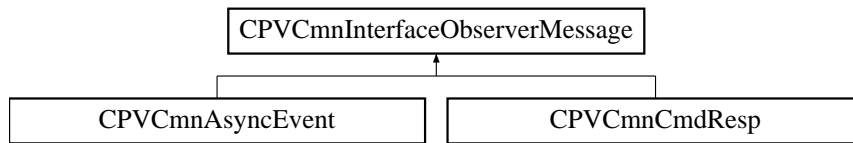
The documentation for this class was generated from the following file:

- [pv_interface_cmd_message.h](#)

4.4 CPVcmmInterfaceObserverMessage Class Reference

```
#include <pv_common_types.h>
```

Inheritance diagram for CPVcmmInterfaceObserverMessage:



Public Member Functions

- [CPVcmmInterfaceObserverMessage \(\)](#)
- [CPVcmmInterfaceObserverMessage \(TPVcmmResponseType aResponseType\)](#)
- [virtual ~CPVcmmInterfaceObserverMessage \(\)](#)
- [TPVcmmResponseType GetResponseType \(\) const](#)
- [virtual int GetPriority \(\) const](#)

Data Fields

- [TPVcmmResponseType iResponseType](#)
- [int iPriority](#)
- [int iOrder](#)

4.4.1 Detailed Description

[CPVcmmInterfaceObserverMessage](#) Class

[CPVcmmInterfaceObserverMessage](#) is the interface to the pv2way SDK, which allows initialization, control, and termination of a two-way 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

4.4.2 Constructor & Destructor Documentation

4.4.2.1 CPVCmnInterfaceObserverMessage::CPVCmnInterfaceObserverMessage () [inline]

4.4.2.2 CPVCmnInterfaceObserverMessage::CPVCmnInterfaceObserverMessage (TPVCmnResponseType *aResponseType*) [inline]

4.4.2.3 virtual CPVCmnInterfaceObserverMessage::~~CPVCmnInterfaceObserverMessage () [inline, virtual]

4.4.3 Member Function Documentation

4.4.3.1 virtual int CPVCmnInterfaceObserverMessage::GetPriority () const [inline, virtual]

References iPriority.

Referenced by CPVCmnInterfaceObserverMessageCompare::compare().

4.4.3.2 TPVCmnResponseType CPVCmnInterfaceObserverMessage::GetResponseType () const [inline]

References iResponseType.

4.4.4 Field Documentation

4.4.4.1 int CPVCmnInterfaceObserverMessage::iOrder

Referenced by CPVCmnInterfaceObserverMessageCompare::compare().

4.4.4.2 int CPVCmnInterfaceObserverMessage::iPriority

Referenced by GetPriority().

4.4.4.3 TPVCmnResponseType CPVCmnInterfaceObserverMessage::iResponseType

Referenced by GetResponseType().

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

- [pv_common_types.h](#)

4.5 CPVCmnInterfaceObserverMessageCompare Class Reference

```
#include <pv_common_types.h>
```

Public Member Functions

- `int compare (CPVCmnInterfaceObserverMessage *a, CPVCmnInterfaceObserverMessage *b) const`

4.5.1 Member Function Documentation

4.5.1.1 `int CPVCmnInterfaceObserverMessageCompare::compare (CPVCmnInterfaceObserverMessage * a, CPVCmnInterfaceObserverMessage * b) const`
[`inline`]

References `CPVCmnInterfaceObserverMessage::GetPriority()`, and `CPVCmnInterfaceObserverMessage::iOrder`.

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

- [pv_common_types.h](#)

4.6 MPVCmnCmdStatusObserver Class Reference

```
#include <pv_common_types.h>
```

Public Member Functions

- virtual `~MPVCmnCmdStatusObserver ()`
- virtual void `CommandCompletedL (const CPVCmnCmdResp &aResponse)=0`

4.6.1 Constructor & Destructor Documentation

4.6.1.1 virtual `MPVCmnCmdStatusObserver::~MPVCmnCmdStatusObserver ()` [`inline`, `virtual`]

4.6.2 Member Function Documentation

4.6.2.1 virtual void `MPVCmnCmdStatusObserver::CommandCompletedL (const CPVCmnCmdResp &aResponse)` [`pure virtual`]

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

- [pv_common_types.h](#)

4.7 MPVCmnErrorEventObserver Class Reference

```
#include <pv_common_types.h>
```

Public Member Functions

- virtual `~MPVCmnErrorEventObserver ()`
- virtual void `HandleErrorEventL (const CPVCmnAsyncErrorEvent &aEvent)=0`

4.7.1 Constructor & Destructor Documentation

4.7.1.1 virtual `MPVCmnErrorEventObserver::~MPVCmnErrorEventObserver ()` [`inline`, `virtual`]

4.7.2 Member Function Documentation

4.7.2.1 virtual void `MPVCmnErrorEventObserver::HandleErrorEventL (const CPVCmnAsyncErrorEvent &aEvent)` [`pure virtual`]

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

- [pv_common_types.h](#)

4.8 MPVCmnInfoEventObserver Class Reference

```
#include <pv_common_types.h>
```

Public Member Functions

- virtual [~MPVCmnInfoEventObserver](#) ()
- virtual void [HandleInformationalEventL](#) (const [CPVCmnAsyncInfoEvent](#) &aEvent)=0

4.8.1 Constructor & Destructor Documentation

4.8.1.1 virtual [MPVCmnInfoEventObserver::~MPVCmnInfoEventObserver](#) () [[inline](#), [virtual](#)]

4.8.2 Member Function Documentation

4.8.2.1 virtual void [MPVCmnInfoEventObserver::HandleInformationalEventL](#) (const [CPVCmnAsyncInfoEvent](#) &*aEvent*) [[pure virtual](#)]

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

- [pv_common_types.h](#)

4.9 PVAsyncErrorEvent Class Reference

```
#include <pv_engine_observer_message.h>
```

Public Member Functions

- [PVAsyncErrorEvent](#) ([PVEventType](#) aEventType, [PVExclusivePtr](#) aEventData=NULL, uint8 *aLocalBuffer=NULL, int32 aLocalBufferSize=0)
- [PVAsyncErrorEvent](#) ([PVEventType](#) aEventType, [OsclAny](#) *aContext, [PVInterface](#) *aEventExtInterface, [PVExclusivePtr](#) aEventData=NULL, uint8 *aLocalBuffer=NULL, int32 aLocalBufferSize=0)
- [~PVAsyncErrorEvent](#) ()
- [PVResponseType](#) [GetResponseType](#) () const
- [PVEventType](#) [GetEventType](#) () const
- void [GetEventData](#) ([PVExclusivePtr](#) &aPtr) const

4.9.1 Detailed Description

[PVAsyncErrorEvent](#) Class

[PVAsyncErrorEvent](#) is used to pass unsolicited error indications to the user. Additional information can be tagged based on the specific event

4.9.2 Constructor & Destructor Documentation

4.9.2.1 [PVAsyncErrorEvent::PVAsyncErrorEvent](#) ([PVEventType](#) *aEventType*, [PVExclusivePtr](#) *aEventData* = NULL, uint8 * *aLocalBuffer* = NULL, int32 *aLocalBufferSize* = 0) **[inline]**

Constructor for [PVAsyncErrorEvent](#)

4.9.2.2 [PVAsyncErrorEvent::PVAsyncErrorEvent](#) ([PVEventType](#) *aEventType*, [OsclAny](#) * *aContext*, [PVInterface](#) * *aEventExtInterface*, [PVExclusivePtr](#) *aEventData* = NULL, uint8 * *aLocalBuffer* = NULL, int32 *aLocalBufferSize* = 0) **[inline]**

Constructor with context and event extension interface

4.9.2.3 [PVAsyncErrorEvent::~~PVAsyncErrorEvent](#) () **[inline]**

Destructor

4.9.3 Member Function Documentation

4.9.3.1 void [PVAsyncErrorEvent::GetEventData](#) ([PVExclusivePtr](#) &*aPtr*) const **[inline]**

Returns

Returns the opaque data associated with the event.

4.9.3.2 PVEventType PVAsyncErrorEvent::GetEventType () const [inline]**Returns**

Returns the Event type that has been received

4.9.3.3 PVResponseType PVAsyncErrorEvent::GetResponseType () const [inline]

WILL BE DEPRECATED SINCE IT IS NOT BEING USED. CURRENTLY RETURNING 0.

Returns

Returns the type of Response we get

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

- [pv_engine_observer_message.h](#)

4.10 PVAsyncInformationalEvent Class Reference

```
#include <pv_engine_observer_message.h>
```

Public Member Functions

- [PVAsyncInformationalEvent](#) ([PVEventType](#) aEventType, [PVExclusivePtr](#) aEventData=NULL, uint8 *aLocalBuffer=NULL, int32 aLocalBufferSize=0)
- [PVAsyncInformationalEvent](#) ([PVEventType](#) aEventType, [OsciAny](#) *aContext, [PVInterface](#) *aEventExtInterface, [PVExclusivePtr](#) aEventData=NULL, uint8 *aLocalBuffer=NULL, int32 aLocalBufferSize=0)
- [~PVAsyncInformationalEvent](#) ()
- [PVResponseType](#) [GetResponseType](#) () const
- [PVEventType](#) [GetEventType](#) () const
- void [GetEventData](#) ([PVExclusivePtr](#) &aPtr) const

4.10.1 Detailed Description

[PVAsyncInformationalEvent](#) Class

[PVAsyncInformationalEvent](#) is used to pass unsolicited informational indications to the user. Additional information can be tagged based on the specific event

4.10.2 Constructor & Destructor Documentation

4.10.2.1 [PVAsyncInformationalEvent::PVAsyncInformationalEvent](#) ([PVEventType](#) aEventType, [PVExclusivePtr](#) aEventData = NULL, uint8 * aLocalBuffer = NULL, int32 aLocalBufferSize = 0) [[inline](#)]

Constructor for [PVAsyncInformationalEvent](#)

4.10.2.2 [PVAsyncInformationalEvent::PVAsyncInformationalEvent](#) ([PVEventType](#) aEventType, [OsciAny](#) * aContext, [PVInterface](#) * aEventExtInterface, [PVExclusivePtr](#) aEventData = NULL, uint8 * aLocalBuffer = NULL, int32 aLocalBufferSize = 0) [[inline](#)]

Constructor with context and event extension interface

4.10.2.3 [PVAsyncInformationalEvent::~~PVAsyncInformationalEvent](#) () [[inline](#)]

Destructor

4.10.3 Member Function Documentation

4.10.3.1 void [PVAsyncInformationalEvent::GetEventData](#) ([PVExclusivePtr](#) & aPtr) const [[inline](#)]

Returns

Returns the opaque data associated with the event.

4.10.3.2 PVEventType PVAsyncInformationalEvent::GetEventType () const [inline]**Returns**

Returns the Event type that has been received

4.10.3.3 PVResponseType PVAsyncInformationalEvent::GetResponseType () const [inline]

WILL BE DEPRECATED SINCE IT IS NOT BEING USED. CURRENTLY RETURNING 0.

Returns

Returns the type of Response we get

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

- [pv_engine_observer_message.h](#)

4.11 PVAuthorEngineFactory Class Reference

```
#include <pvauthorenginefactory.h>
```

Static Public Member Functions

- static OSCL_IMPORT_REF PVAuthorEngineInterface * CreateAuthor (PVCommandStatusObserver *aCmdStatusObserver, PVErrorEventObserver *aErrorEventObserver, PVInformationalEventObserver *aInfoEventObserver)
- static OSCL_IMPORT_REF bool DeleteAuthor (PVAuthorEngineInterface *aAuthor)

4.11.1 Detailed Description

PVAuthorEngineFactory Class

PVAuthorEngineFactory class is a singleton class which instantiates and provides access to pvAuthor engine. It returns an PVAuthorEngineInterface reference, the interface class of the pvAuthor SDK.

The application is expected to contain and maintain a pointer to the PVAuthorEngineInterface instance at all time that pvAuthor engine is active.

4.11.2 Member Function Documentation

4.11.2.1 static OSCL_IMPORT_REF PVAuthorEngineInterface* PVAuthorEngineFactory::CreateAuthor (PVCommandStatusObserver * aCmdStatusObserver, PVErrorEventObserver * aErrorEventObserver, PVInformationalEventObserver * aInfoEventObserver) [static]

Creates an instance of a pvAuthor engine. If the creation fails, this function will leave.

Parameters

- aCmdStatusObserver* The observer for command status
- aErrorEventObserver* The observer for unsolicited error events
- aInfoEventObserver* The observer for unsolicited informational events

Returns

A pointer to an author or leaves if instantiation fails

4.11.2.2 static OSCL_IMPORT_REF bool PVAuthorEngineFactory::DeleteAuthor (PVAuthorEngineInterface * aAuthor) [static]

This function allows the application to delete an instance of a pvAuthor and reclaim all allocated resources. An author can be deleted only in the idle state. An attempt to delete an author in any other state will fail and return false.

Parameters

- aAuthor* The author to be deleted.

Returns

A status code indicating success or failure.

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

- [pvauthorenginefactory.h](#)

4.12 PVAuthorEngineInterface Class Reference

```
#include <pvauthorengineinterface.h>
```

Public Member Functions

- virtual [~PVAuthorEngineInterface](#) ()
- virtual [PVCommandId SetLogAppender](#) (const char *aTag, PVLoggerAppender &aAppender, const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId RemoveLogAppender](#) (const char *aTag, PVLoggerAppender &aAppender, const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId SetLogLevel](#) (const char *aTag, int32 aLevel, bool aSetSubtree=false, const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId GetLogLevel](#) (const char *aTag, [PVLogLevelInfo](#) &aLogInfo, const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId Open](#) (const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId Close](#) (const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId AddDataSource](#) (const PVMFNNodeInterface &aDataSource, const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId RemoveDataSource](#) (const PVMFNNodeInterface &aDataSource, const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId SelectComposer](#) (const PvmfMimeType &aComposerType, PVInterface *&aConfigInterface, const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId SelectComposer](#) (const PVUuid &aComposerUuid, PVInterface *&aConfigInterface, const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId AddMediaTrack](#) (const PVMFNNodeInterface &aDataSource, const PvmfMimeType &aEncoderType, const OsciAny *aComposer, PVInterface *&aConfigInterface, const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId AddMediaTrack](#) (const PVMFNNodeInterface &aDataSource, const PVUuid &aEncoderUuid, const OsciAny *aComposer, PVInterface *&aConfigInterface, const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId AddDataSink](#) (const PVMFNNodeInterface &aDataSink, const OsciAny *aComposer, const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId RemoveDataSink](#) (const PVMFNNodeInterface &aDataSink, const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId Init](#) (const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId Reset](#) (const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId Start](#) (const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId Pause](#) (const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId Resume](#) (const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId Stop](#) (const OsciAny *aContextData=NULL)=0
- virtual [PVAEState GetPVAuthorState](#) ()=0
- virtual [PVCommandId QueryInterface](#) (const PVUuid &aUuid, PVInterface *&aInterfacePtr, const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId GetSDKModuleInfo](#) ([PVSDKModuleInfo](#) &aSDKModuleInfo, const OsciAny *aContextData=NULL)=0
- virtual [PVCommandId CancelAllCommands](#) (const OsciAny *aContextData=NULL)=0

Static Public Member Functions

- static OSCL_IMPORT_REF void [GetSDKInfo](#) ([PVSDKInfo](#) &aSDKInfo)

4.12.1 Detailed Description

[PVAuthorEngineInterface](#)

4.12.2 Constructor & Destructor Documentation

4.12.2.1 `virtual PVAuthorEngineInterface::~PVAuthorEngineInterface () [inline, virtual]`

Destructor.

4.12.3 Member Function Documentation

4.12.3.1 `virtual PVCommandId PVAuthorEngineInterface::AddDataSink (const PVMFNodeInterface & aDataSink, const OsclAny * aComposer, const OsclAny * aContextData = NULL) [pure virtual]`

Adds a media sink where output data from the specified composer will be written to. Currently this API does not cause any action as it is not relevant.

This command is valid only when pvAuthor Engine is in PVAE_STATE_OPENED state. The referenced composer must be previously selected.

This command does not change the pvAuthor Engine engine state.

Parameters

aDataSink Reference to the data sink to be used

aComposer Opaque data identifying the composer to which the data sink will connect to.

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

Returns

A unique command id for asynchronous completion

4.12.3.2 `virtual PVCommandId PVAuthorEngineInterface::AddDataSource (const PVMFNodeInterface & aDataSource, const OsclAny * aContextData = NULL) [pure virtual]`

Adds a media source to be used as input to an authoring session.

This command is valid only when pvAuthor Engine is in PVAE_STATE_OPENED state. This command does not change the pvAuthor Engine engine state.

Parameters

aDataSource Reference to the data source

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

Returns

Unique command ID to identify this command in command response

4.12.3.3 virtual PVCommandId PVAuthorEngineInterface::AddMediaTrack (const PVMFNodeInterface & aDataSource, const PVUuid & aEncoderUuid, const OsclAny * aComposer, PVInterface *& aConfigInterface, const OsclAny * aContextData = NULL) [pure virtual]

Add a media track to the specified composer.

The source data of this media track will come from the specified data source. pvAuthor engine will encoder of the specified Uuid to encode the source data. A media track will be added to the specified composer, and encoded data will be written to the composer during the authoring session.

A configuration object for the selected composer will be saved to the PVInterface pointer provided in aConfigInterface parameter. User should call queryInterface to query for the configuration interfaces supported by the encoder. Before calling [Reset\(\)](#), user must call removeRef on the PVInterface object to remove its reference to the object.

This command is valid only when pvAuthor Engine is in PVAE_STATE_OPENED state. The referenced data source and composer must be already added before this method is called. This command does not change the pvAuthor Engine engine state.

Parameters

aDataSource Data source node to provide input data

aEncoderUuid Uuid of encoder to encode the source data

aComposer Opaque data to identify the composer in which a media track will be added.

aConfigInterface Pointer to configuration object for the selected encoder will be saved to this parameter upon completion of this call

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

Returns

A unique command id for asynchronous completion

4.12.3.4 virtual PVCommandId PVAuthorEngineInterface::AddMediaTrack (const PVMFNodeInterface & aDataSource, const PvmfMimeType & aEncoderType, const OsclAny * aComposer, PVInterface *& aConfigInterface, const OsclAny * aContextData = NULL) [pure virtual]

Add a media track to the specified composer.

The source data of this media track will come from the specified data source. pvAuthor engine will select the most suitable available encoder of the specified type. A media track will be added to the specified composer, and encoded data will be written to the composer during the authoring session.

A configuration object for the selected composer will be saved to the PVInterface pointer provided in aConfigInterface parameter. User should call queryInterface to query for the configuration interfaces supported by the encoder. Before calling [Reset\(\)](#), user must call removeRef on the PVInterface object to remove its reference to the object.

This command is valid only when pvAuthor Engine is in PVAE_STATE_OPENED state. The referenced data source and composer must be already added before this method is called. This command does not change the pvAuthor Engine engine state.

Parameters

aDataSource Data source node to provide input data

aEncoderType MIME type of encoder to encode the source data

aComposer Opaque data to identify the composer in which a media track will be added.

aConfigInterface Pointer to configuration object for the selected encoder will be saved to this parameter upon completion of this call

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

Returns

A unique command id for asynchronous completion

4.12.3.5 virtual PVCommandId PVAuthorEngineInterface::CancelAllCommands (const OsclAny * aContextData = NULL) [pure virtual]

Cancel all pending requests. The current request being processed, if any, will also be aborted. PVAE_CMD_CANCEL_ALL_COMMANDS will be passed to the command observer on completion. Currently this API is NOT SUPPORTED.

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

4.12.3.6 virtual PVCommandId PVAuthorEngineInterface::Close (const OsclAny * aContextData = NULL) [pure virtual]

Closes an authoring session.

All resources added and allocated to the authoring session will be released.

This command is valid only when pvAuthor engine is in PVAE_STATE_OPENED state and Upon completion of this command, pvAuthor Engine will be in PVAE_STATE_IDLE state.

Parameters

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

Returns

Unique command ID to identify this command in command response

4.12.3.7 virtual PVCommandId PVAuthorEngineInterface::GetLogLevel (const char * aTag, PVLogLevelInfo & aLogInfo, const 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). Currently this API is NOT SUPPORTED.

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

4.12.3.8 virtual PVAEState PVAuthorEngineInterface::GetPVAuthorState () [pure virtual]

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

Parameters

aState Output parameter to hold state information

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

Returns

A unique command id for synchronous completion

4.12.3.9 static OSCL_IMPORT_REF void PVAuthorEngineInterface::GetSDKInfo (PVSDKInfo & aSDKInfo) [static]

Returns SDK version information about author engine.

Parameters

aSDKInfo A reference to a [PVSDKInfo](#) structure which contains product name, supported hardware platform, supported software platform, version, part number, and PV UUID. These fields will contain info .for the currently instantiated pvPlayer engine when this function returns success.

4.12.3.10 virtual PVCommandId PVAuthorEngineInterface::GetSDKModuleInfo (PVSDKModuleInfo & aSDKModuleInfo, const OsclAny * aContextData = NULL) [pure virtual]

Returns information about all modules currently used by the SDK. Currently this API is NOT SUPPORTED.

Parameters

aSDKModuleInfo A reference to a PVSDKModuleInfo structure which contains the number of modules currently used by pvAuthor Engine 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

Returns

A unique command id for asynchronous completion

4.12.3.11 virtual PVCommandId PVAuthorEngineInterface::Init (const OsclAny * *aContextData* = NULL) [pure virtual]

Initialize an authoring session.

Upon calling this method, no more data sources and sinks can be added to the session. Also, all configuration settings will be locked and cannot be modified until the session is reset by calling [Reset\(\)](#). Resources for the session will be allocated and initialized to the configuration settings specified. This command is valid only when pvAuthor Engine is in PVAE_STATE_OPENED state.

Upon completion of this command, pvAuthor Engine will be in PVAE_STATE_INITIALIZED state, and the authoring session is ready to start.

Parameters

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

Returns

A unique command id for asynchronous completion

4.12.3.12 virtual PVCommandId PVAuthorEngineInterface::Open (const OsclAny * *aContextData* = NULL) [pure virtual]

Opens an authoring session.

This command is valid only when pvAuthor engine is in PVAE_STATE_IDLE state. Upon completion of this method, pvAuthor engine will be in PVAE_STATE_OPENED state.

Parameters

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

Returns

Unique command ID to identify this command in command response

4.12.3.13 virtual PVCommandId PVAuthorEngineInterface::Pause (const OsclAny * *aContextData* = NULL) [pure virtual]

Pause the authoring session.

The authoring session will be paused and no encoded output data will be sent to the data sink. This function is valid only in the PVAE_STATE_RECORDING state.

Upon completion of this command, pvAuthor Engine will be in PVAE_STATE_PAUSED state.

Parameters

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

Returns

A unique command id for asynchronous completion

4.12.3.14 `virtual PVCommandId PVAuthorEngineInterface::QueryInterface (const PVUuid & aUuid, PVInterface *& aInterfacePtr, const OsclAny * aContextData = NULL) [pure virtual]`

This API is to allow for extensibility of the pvAuthor engine 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

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

Returns

A unique command id for asynchronous completion

4.12.3.15 `virtual PVCommandId PVAuthorEngineInterface::RemoveDataSink (const PVMFNNodeInterface & aDataSink, const OsclAny * aContextData = NULL) [pure virtual]`

Removes a previously added data sink. Currently this API does not cause any action as it is not relevant.

This command is valid only when pvAuthor Engine is in PVAE_STATE_OPENED state. This command does not change the pvAuthor Engine engine state.

Parameters

aDataSink Reference to the data sink to be removed

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

Returns

A unique command id for asynchronous completion

4.12.3.16 `virtual PVCommandId PVAuthorEngineInterface::RemoveDataSource (const PVMFNNodeInterface & aDataSource, const OsclAny * aContextData = NULL) [pure virtual]`

Unbinds a previously added data source.

This command is valid only when pvAuthor Engine is in PVAE_STATE_OPENED state. This command does not change the pvAuthor Engine engine state.

Parameters

aDataSource Reference to the data source to be removed

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

Returns

A unique command id for asynchronous completion

4.12.3.17 virtual PVCommandId PVAuthorEngineInterface::RemoveLogAppender (const char * aTag, PVLoggerAppender & aAppender, const OsclAny * aContextData = NULL) [pure virtual]

Allows a logging appender to be removed from the logger tree at the point specified by the input tag. If the input tag is NULL then the appender will be removed from locations in the tree. Currently this API is NOT SUPPORTED.

Parameters

aTag Specifies the logger tree tag where the appender should be removed. Can be NULL to remove at all locations.

aAppender The log appender to remove.

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

4.12.3.18 virtual PVCommandId PVAuthorEngineInterface::Reset (const OsclAny * aContextData = NULL) [pure virtual]

Reset an initialized authoring session.

The authoring session will be stopped and all composers and encoders selected for the session will be removed. All data sources and sinks will be reset but will continue to be available for authoring the next output clip.

User must call removeRef() to remove its reference to any PVInterface objects received from [SelectComposer\(\)](#) or [AddMediaTrack\(\)](#) or [QueryInterface\(\)](#) APIs before calling this method. This method would fail otherwise.

This method can be called from ANY state but PVAE_STATE_IDLE. Upon completion of this command, pvAuthor Engine will be in PVAE_STATE_OPENED state.

Parameters

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

Returns

A unique command id for asynchronous completion

4.12.3.19 virtual PVCommandId PVAuthorEngineInterface::Resume (const OsclAny * *aContextData* = NULL) [pure virtual]

Resume a paused authoring session.

The authoring session will be resumed and pvAuthor Engine will resume sending encoded output data to the data sinks. This function is valid only in the PVAE_STATE_PAUSED state.

Upon completion of this command, pvAuthor Engine will be in PVAE_STATE_RECORDING state.

Parameters

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

Returns

A unique command id for asynchronous completion

4.12.3.20 virtual PVCommandId PVAuthorEngineInterface::SelectComposer (const PVUuid & *aComposerUuid*, PVInterface *& *aConfigInterface*, const OsclAny * *aContextData* = NULL) [pure virtual]

Selects an output composer by specifying its Uuid.

pvAuthor engine the composer of the specified Uuid in the authoring session. This command is valid only when pvAuthor Engine is in PVAE_STATE_OPENED state. This command does not change the pvAuthor Engine state.

Upon completion of this command, opaque data to identify the selected composer is provided in the call-back. The user needs to use this opaque data to identify the composer when calling [AddMediaTrack\(\)](#), [AddDataSink\(\)](#). A configuration interface for the selected composer will be saved to the PVInterface pointer provided in aConfigInterface parameter. User should call queryInterface to query for the configuration interfaces supported by the composer. When configuration is complete or before calling [Reset\(\)](#), user must call removeRef on the PVInterface object to remove its reference to the object.

Parameters

aComposerUuid Uuid of output composer to be used

aConfigInterface Pointer to configuration object for the selected composer will be saved to this parameter upon completion of this call

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

Returns

A unique command id for asynchronous completion

4.12.3.21 virtual PVCommandId PVAuthorEngineInterface::SelectComposer (const PvmfMimeString & *aComposerType*, PVInterface *& *aConfigInterface*, const OsclAny * *aContextData* = NULL) [pure virtual]

Selects an output composer by specifying its MIME type.

pvAuthor engine will use the most suitable output composer of the specified MIME type available in the authoring session. This command is valid only when pvAuthor Engine is in PVAE_STATE_OPENED state. This command does not change the pvAuthor Engine state.

Upon completion of this command, opaque data to identify the selected composer is provided in the callback. The user needs to use this opaque data to identify the composer when calling `AddMediaTrack()`, `AddDataSink()`. A configuration interface for the selected composer will be saved to the `PVInterface` pointer provided in `aConfigInterface` parameter. User should call `queryInterface` to query for the configuration interfaces supported by the composer. When configuration is complete or before calling `Reset()`, user must call `removeRef` on the `PVInterface` object to remove its reference to the object.

Parameters

- aComposerType* MIME type of output composer to be used
- aConfigInterface* Pointer to configuration object for the selected composer will be saved to this parameter upon completion of this call
- aContextData* Optional opaque data to be passed back to user with the command response

Returns

A unique command id for asynchronous completion

4.12.3.22 virtual PVCommandId PVAuthorEngineInterface::SetLogAppender (const char * aTag, PVLoggerAppender & aAppender, const 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). Currently this API is NOT SUPPORTED.

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

4.12.3.23 virtual PVCommandId PVAuthorEngineInterface::SetLogLevel (const char * aTag, int32 aLevel, bool aSetSubtree = false, const 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. Currently this API is NOT SUPPORTED.

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

4.12.3.24 **virtual PVCommandId PVAuthorEngineInterface::Start (const OsclAny * aContextData = NULL) [pure virtual]**

Start the authoring session.

pvAuthor Engine will begin to receive source data, encode them to the specified format and quality, and send the output data to the specified data sinks. This function is valid only in the PVAE_STATE_INITIALIZED state.

Upon completion of this command, pvAuthor Engine will be in PVAE_STATE_RECORDING state.

Parameters

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

Returns

A unique command id for asynchronous completion

4.12.3.25 **virtual PVCommandId PVAuthorEngineInterface::Stop (const OsclAny * aContextData = NULL) [pure virtual]**

Stops an authoring session.

The authoring session will be stopped and pvAuthor Engine will stop receiving source data from the data sources, and no further encoded data will be sent to the data sinks. This function is valid only in the PVAE_STATE_RECORDING and PVAE_STATE_PAUSED states.

Upon completion of this command, pvAuthor Engine will be in PVAE_STATE_INITIALIZED state.

Parameters

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

Returns

A unique command id for asynchronous completion

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

- [pvauthorengineinterface.h](#)

4.13 PVCmdResponse Class Reference

```
#include <pv_engine_observer_message.h>
```

Public Member Functions

- [PVCmdResponse](#) ([PVCCommandId](#) aId, [OsclAny](#) *aContext, [PVMFStatus](#) aStatus, [OsclAny](#) *aEventData=NULL, [int32](#) aEventDataSize=0)
- [PVCmdResponse](#) ([PVCCommandId](#) aId, [OsclAny](#) *aContext, [PVMFStatus](#) aStatus, [PVInterface](#) *aEventExtInterface=NULL, [OsclAny](#) *aEventData=NULL, [int32](#) aEventDataSize=0)
- [PVResponseType](#) [GetResponseType](#) () const
- [PVCCommandId](#) [GetCmdId](#) () const
- [OsclAny](#) * [GetContext](#) () const
- [PVMFStatus](#) [GetCmdStatus](#) () const
- [OsclAny](#) * [GetResponseData](#) () const
- [int32](#) [GetResponseDataSize](#) () const
- [PVMFStatus](#) [GetExtendedErrorMessage](#) (const [PVUuid](#) &auuid, [PVInterface](#) *&aface) const

4.13.1 Detailed Description

[PVCmdResponse](#) Class

[PVCmdResponse](#) class is used to pass completion status on previously issued commands

4.13.2 Constructor & Destructor Documentation

4.13.2.1 [PVCmdResponse::PVCmdResponse](#) ([PVCCommandId](#) aId, [OsclAny](#) * aContext, [PVMFStatus](#) aStatus, [OsclAny](#) * aEventData = NULL, [int32](#) aEventDataSize = 0) **[inline]**

Constructor for [PVCmdResponse](#)

4.13.2.2 [PVCmdResponse::PVCmdResponse](#) ([PVCCommandId](#) aId, [OsclAny](#) * aContext, [PVMFStatus](#) aStatus, [PVInterface](#) * aEventExtInterface = NULL, [OsclAny](#) * aEventData = NULL, [int32](#) aEventDataSize = 0) **[inline]**

Constructor with event extension interface

4.13.3 Member Function Documentation

4.13.3.1 [PVCCommandId](#) [PVCmdResponse::GetCmdId](#) () const **[inline]**

Returns

Returns the unique ID associated with a command of this type.

4.13.3.2 PVMFStatus PVCmdResponse::GetCmdStatus () const [inline]**Returns**

Returns the completion status of the command

4.13.3.3 OsciAny* PVCmdResponse::GetContext () const [inline]**Returns**

Returns the opaque data that was passed in with the command.

4.13.3.4 PVMFStatus PVCmdResponse::GetExtendedErrorMessage (const PVUuid & auuid, PVInterface *& aface) const [inline]**4.13.3.5 OsciAny* PVCmdResponse::GetResponseData () const [inline]**

WILL BE DEPRECATED WHEN PVMFCmdResp REMOVES EVENT DATA

Returns

Returns additional data associated with the command. This is to be interpreted based on the command issued and the return status

4.13.3.6 int32 PVCmdResponse::GetResponseDataSize () const [inline]**4.13.3.7 PVResponseType PVCmdResponse::GetResponseType () const [inline]**

WILL BE DEPRECATED SINCE IT IS NOT BEING USED. CURRENTLY RETURNS 0

Returns

Returns the type of Response we get

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

- [pv_engine_observer_message.h](#)

4.14 PVCommandStatusObserver Class Reference

```
#include <pv_engine_observer.h>
```

Public Member Functions

- virtual void [CommandCompleted](#) (const [PVCmdResponse](#) &aResponse)=0
- virtual [~PVCommandStatusObserver](#) ()

4.14.1 Detailed Description

[PVCommandStatusObserver](#) Class

[PVCommandStatusObserver](#) is the PV SDK observer class for notifying the status of issued command messages. The API provides a mechanism for the status of each command to be passed back along with context specific information where applicable. Applications using the PV SDKs must have a class derived from [PVCommandStatusObserver](#) and implement the pure virtual function in order to receive event notifications from a PV SDK. Additional information is optionally provided via derived classes.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 `virtual PVCommandStatusObserver::~~PVCommandStatusObserver () [inline, virtual]`

4.14.3 Member Function Documentation

4.14.3.1 `virtual void PVCommandStatusObserver::CommandCompleted (const PVCmdResponse &aResponse) [pure virtual]`

Handle an event that has been generated.

Parameters

aResponse "The response to a previously issued command."

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

- [pv_engine_observer.h](#)

4.15 PVConfigInterface Class Reference

```
#include <pv_config_interface.h>
```

4.15.1 Detailed Description

Base interface for all configuration classes

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

- [pv_config_interface.h](#)

4.16 PVEngineAsyncEvent Class Reference

```
#include <pv_engine_types.h>
```

Public Member Functions

- [PVEngineAsyncEvent](#) (int32 aAsyncEventType)
- [PVEngineAsyncEvent](#) (const [PVEngineAsyncEvent](#) &aAsyncEvent)
- int32 [GetAsyncEventType](#) () const

Data Fields

- int32 [iAsyncEventType](#)

4.16.1 Detailed Description

[PVEngineAsyncEvent](#) Class

[PVEngineAsyncEvent](#) class is a data class to hold asynchronous events generated by the engine. The class is meant to be used inside the engine and not exposed to the interface layer or above.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 PVEngineAsyncEvent::PVEngineAsyncEvent (int32 aAsyncEventType) [inline]

The constructor for [PVEngineCommand](#) which allows the data values to be set.

Parameters

aCmdType The command type value for this command. The value is an engine-specific 32-bit value.

aCmdId The command ID assigned by the engine for this command.

aContextData The pointer to the passed-in context data for this command.

Returns

None

4.16.2.2 PVEngineAsyncEvent::PVEngineAsyncEvent (const PVEngineAsyncEvent &aAsyncEvent) [inline]

The copy constructor for [PVEngineAsyncEvent](#). Used mainly for `Oscl_Vector`.

Parameters

aAsyncEvent The reference to the source [PVEngineAsyncEvent](#) to copy the data values from.

Returns

None

References [iAsyncEventType](#).

4.16.3 Member Function Documentation

4.16.3.1 `int32 PVEngineAsyncEvent::GetAsyncEventType () const [inline]`

This function returns the stored asynchronous event type value.

Returns

The signed 32-bit event type value.

References `iAsyncEventType`.

4.16.4 Field Documentation

4.16.4.1 `int32 PVEngineAsyncEvent::iAsyncEventType`

Referenced by `GetAsyncEventType()`, and `PVEngineAsyncEvent()`.

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

- [pv_engine_types.h](#)

4.17 PVEngineCommand Class Reference

```
#include <pv_engine_types.h>
```

Public Member Functions

- [PVEngineCommand](#) (int32 aCmdType, [PVCommandId](#) aCmdId, OsclAny *aContextData=NULL, OsclAny *aParam1=NULL, OsclAny *aParam2=NULL, OsclAny *aParam3=NULL)
- [PVEngineCommand](#) (const [PVEngineCommand](#) &aCmd)
- int32 [GetCmdType](#) () const
- [PVCommandId](#) [GetCmdId](#) () const
- OsclAny * [GetContext](#) () const
- OsclAny * [GetParam1](#) () const
- OsclAny * [GetParam2](#) () const
- OsclAny * [GetParam3](#) () const
- const PvmfMimeString & [GetMimeType](#) () const
- PVUuid [GetUuid](#) () const
- void [SetMimeType](#) (const PvmfMimeString &aMimeType)
- void [SetUuid](#) (const PVUuid &aUuid)

Data Fields

- int32 [iCmdType](#)
- [PVCommandId](#) [iCmdId](#)
- OsclAny * [iContextData](#)
- OsclAny * [iParam1](#)
- OsclAny * [iParam2](#)
- OsclAny * [iParam3](#)
- OSCL_HeapString< OsclMemAllocator > [iMimeType](#)
- PVUuid [iUuid](#)

4.17.1 Detailed Description

[PVEngineCommand](#) Class

[PVEngineCommand](#) class is a data class to hold issued commands. The class is meant to be used inside the engine and not exposed to the interface layer or above.

4.17.2 Constructor & Destructor Documentation

- 4.17.2.1 [PVEngineCommand::PVEngineCommand](#) (int32 *aCmdType*, [PVCommandId](#) *aCmdId*, OsclAny * *aContextData* = NULL, OsclAny * *aParam1* = NULL, OsclAny * *aParam2* = NULL, OsclAny * *aParam3* = NULL) [inline]**

The constructor for [PVEngineCommand](#) which allows the data values to be set.

Parameters

aCmdType The command type value for this command. The value is an engine-specific 32-bit value.

aCmdId The command ID assigned by the engine for this command.

aContextData The pointer to the passed-in context data for this command.

Returns

None

4.17.2.2 **PVEngineCommand::PVEngineCommand (const PVEngineCommand & aCmd) [inline]**

The copy constructor for [PVEngineCommand](#). Used mainly for `OscI_Vector`.

Parameters

aCmd The reference to the source [PVEngineCommand](#) to copy the data values from.

Returns

None

References `iCmdId`, `iCmdType`, `iContextData`, `iMimeType`, `iParam1`, `iParam2`, `iParam3`, and `iUuid`.

4.17.3 Member Function Documentation

4.17.3.1 **PVCommandId PVEngineCommand::GetCmdId () const [inline]**

This function returns the stored command ID value.

Returns

The `PVCommandId` value for this command.

References `iCmdId`.

4.17.3.2 **int32 PVEngineCommand::GetCmdType () const [inline]**

This function returns the stored command type value.

Returns

The signed 32-bit command type value for this command.

References `iCmdType`.

4.17.3.3 **OscIAny* PVEngineCommand::GetContext () const [inline]**

This function returns the stored context data pointer.

Returns

The pointer to the context data for this command

References `iContextData`.

4.17.3.4 const PvmfMimeString& PVEngineCommand::GetMimeType () const [inline]

This function returns Mime type parameter for this command

Returns

The Mime type parameter for this command

References iMimeType.

4.17.3.5 OsclAny* PVEngineCommand::GetParam1 () const [inline]

This function returns the first stored parameter pointer.

Returns

The pointer to the first stored parameter for this command

References iParam1.

4.17.3.6 OsclAny* PVEngineCommand::GetParam2 () const [inline]

This function returns the second stored parameter pointer.

Returns

The pointer to the second stored parameter for this command

References iParam2.

4.17.3.7 OsclAny* PVEngineCommand::GetParam3 () const [inline]

This function returns the third stored parameter pointer.

Returns

The pointer to the third stored parameter for this command

References iParam3.

4.17.3.8 PVUuid PVEngineCommand::GetUuid () const [inline]

This function returns Uuid parameter for this command

Returns

The Uuid parameter for this command

References iUuid.

4.17.3.9 void PVEngineCommand::SetMimeType (const PvmfMimeType & *aMimeType*) [inline]

This function stores Mime type parameter of this command

References iMimeType.

4.17.3.10 void PVEngineCommand::SetUuid (const PVUuid & *aUuid*) [inline]

This function stores the Uuid parameter of this command

References iUuid.

4.17.4 Field Documentation

4.17.4.1 PVCommandId PVEngineCommand::iCmdId

Referenced by GetCmdId(), and PVEngineCommand().

4.17.4.2 int32 PVEngineCommand::iCmdType

Referenced by GetCmdType(), and PVEngineCommand().

4.17.4.3 OsclAny* PVEngineCommand::iContextData

Referenced by GetContext(), and PVEngineCommand().

4.17.4.4 OSCL_HeapString<OsclMemAllocator> PVEngineCommand::iMimeType

Referenced by GetMimeType(), PVEngineCommand(), and SetMimeType().

4.17.4.5 OsclAny* PVEngineCommand::iParam1

Referenced by GetParam1(), and PVEngineCommand().

4.17.4.6 OsclAny* PVEngineCommand::iParam2

Referenced by GetParam2(), and PVEngineCommand().

4.17.4.7 OsclAny* PVEngineCommand::iParam3

Referenced by GetParam3(), and PVEngineCommand().

4.17.4.8 PVUuid PVEngineCommand::iUuid

Referenced by GetUuid(), PVEngineCommand(), and SetUuid().

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

- [pv_engine_types.h](#)

4.18 PVErrrorEventObserver Class Reference

```
#include <pv_engine_observer.h>
```

Public Member Functions

- virtual void [HandleErrorEvent](#) (const [PVAsyncErrorEvent](#) &aEvent)=0
- virtual [~PVErrrorEventObserver](#) ()

4.18.1 Detailed Description

[PVErrrorEventObserver](#) Class

[PVErrrorEventObserver](#) is the PV SDK event observer class. It is used for communicating unsolicited error events back to the user of the SDK.

Applications using the PV SDKs must have a class derived from [PVErrrorEventObserver](#) and implement the pure virtual function in order to receive error notifications from a PV SDK.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 virtual [PVErrrorEventObserver::~PVErrrorEventObserver](#) () [[inline](#), [virtual](#)]

4.18.3 Member Function Documentation

4.18.3.1 virtual void [PVErrrorEventObserver::HandleErrorEvent](#) (const [PVAsyncErrorEvent](#) &*aEvent*) [[pure virtual](#)]

Handle an error event that has been generated.

Parameters

aEvent "The event to be handled."

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

- [pv_engine_observer.h](#)

4.19 PVInformationalEventObserver Class Reference

```
#include <pv_engine_observer.h>
```

Public Member Functions

- virtual void [HandleInformationalEvent](#) (const [PVAsyncInformationalEvent](#) &aEvent)=0
- virtual [~PVInformationalEventObserver](#) ()

4.19.1 Detailed Description

[PVInformationalEventObserver](#) Class

[PVInformationalEventObserver](#) is the PV SDK event observer class. It is used for communicating unsolicited informational events back to the user of the SDK.

Applications using the PV SDKs must have a class derived from [PVInformationalEventObserver](#) and implement the pure virtual function in order to receive informational event notifications from a PV SDK.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 virtual [PVInformationalEventObserver::~PVInformationalEventObserver](#) ()
[inline, virtual]

4.19.3 Member Function Documentation

4.19.3.1 virtual void [PVInformationalEventObserver::HandleInformationalEvent](#) (const [PVAsyncInformationalEvent](#) &*aEvent*) [pure virtual]

Handle an informational event that has been generated.

Parameters

aEvent "The event to be handled."

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

- [pv_engine_observer.h](#)

4.20 PVSDKInfo Struct Reference

```
#include <pv_engine_types.h>
```

Public Member Functions

- [PVSDKInfo \(\)](#)
- [PVSDKInfo & operator= \(const PVSDKInfo &aSDKInfo\)](#)

Data Fields

- [OSCL_StackString< 80 > iLabel](#)
- [uint32 iDate](#)

4.20.1 Constructor & Destructor Documentation

4.20.1.1 PVSDKInfo::PVSDKInfo () `[inline]`

References [iDate](#).

4.20.2 Member Function Documentation

4.20.2.1 PVSDKInfo& PVSDKInfo::operator= (const PVSDKInfo & *aSDKInfo*) `[inline]`

References [iDate](#), and [iLabel](#).

4.20.3 Field Documentation

4.20.3.1 uint32 PVSDKInfo::iDate

Referenced by [operator=\(\)](#), and [PVSDKInfo\(\)](#).

4.20.3.2 OSCL_StackString<80> PVSDKInfo::iLabel

Referenced by [operator=\(\)](#).

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

- [pv_engine_types.h](#)

4.21 TPVCmnSDKInfo Struct Reference

```
#include <pv_common_types.h>
```

Public Member Functions

- [TPVCmnSDKInfo \(\)](#)
- [TPVCmnSDKInfo & operator= \(const TPVCmnSDKInfo &aSDKInfo\)](#)

Data Fields

- [OSCL_StackString< 80 > iLabel](#)
- [uint32 iDate](#)

4.21.1 Constructor & Destructor Documentation

4.21.1.1 TPVCmnSDKInfo::TPVCmnSDKInfo () [inline]

References [iDate](#).

4.21.2 Member Function Documentation

4.21.2.1 TPVCmnSDKInfo& TPVCmnSDKInfo::operator= (const TPVCmnSDKInfo &*aSDKInfo*) [inline]

References [iDate](#), and [iLabel](#).

4.21.3 Field Documentation

4.21.3.1 uint32 TPVCmnSDKInfo::iDate

Referenced by [operator=\(\)](#), and [TPVCmnSDKInfo\(\)](#).

4.21.3.2 OSCL_StackString<80> TPVCmnSDKInfo::iLabel

Referenced by [operator=\(\)](#).

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

- [pv_common_types.h](#)

Chapter 5

File Documentation

5.1 pv_common_types.h File Reference

```
#include "oscl_types.h"  
#include "oscl_mem.h"  
#include "oscl_string_containers.h"
```

Data Structures

- struct [TPVCmnSDKInfo](#)
- class [CPVCmnInterfaceObserverMessage](#)
- class [CPVCmnInterfaceObserverMessageCompare](#)
- class [CPVCmnCmdResp](#)
- class [CPVCmnAsyncEvent](#)
- class [MPVCmnErrorEventObserver](#)
- class [MPVCmnInfoEventObserver](#)
- class [MPVCmnCmdStatusObserver](#)

Defines

- #define [PV_COMMON_ASYNC_EVENT_LOCAL_BUF_SIZE](#) 8

Typedefs

- typedef int32 [TPVCmnCommandType](#)
- typedef int32 [TPVCmnCommandId](#)
- typedef int32 [TPVCmnCommandStatus](#)
- typedef int32 [TPVCmnEventType](#)
- typedef void * [TPVCmnExclusivePtr](#)
- typedef void * [TPVCmnInterfacePtr](#)
- typedef int32 [TPVCmnResponseType](#)
- typedef int32 [TPVCmnSDKModuleInfo](#)
- typedef uint8 * [TPVCmnMIMEType](#)

- typedef uint32 [TPVCmnUUID](#)
- typedef int32 [CPVCmnVideoCaps](#)
- typedef int32 [CPVCmnVideoPrefs](#)
- typedef int32 [CPVCmnAudioCaps](#)
- typedef int32 [CPVCmnAudioPrefs](#)
- typedef [CPVCmnAsyncEvent](#) [CPVCmnAsyncInfoEvent](#)
- typedef [CPVCmnAsyncEvent](#) [CPVCmnAsyncErrorEvent](#)

5.1.1 Define Documentation

5.1.1.1 #define PV_COMMON_ASYNC_EVENT_LOCAL_BUF_SIZE 8

Referenced by [CPVCmnAsyncEvent::CPVCmnAsyncEvent\(\)](#).

5.1.2 Typedef Documentation

5.1.2.1 typedef CPVCmnAsyncEvent CPVCmnAsyncErrorEvent

5.1.2.2 typedef CPVCmnAsyncEvent CPVCmnAsyncInfoEvent

5.1.2.3 typedef int32 CPVCmnAudioCaps

5.1.2.4 typedef int32 CPVCmnAudioPrefs

5.1.2.5 typedef int32 CPVCmnVideoCaps

5.1.2.6 typedef int32 CPVCmnVideoPrefs

5.1.2.7 typedef int32 TPVCmnCommandId

5.1.2.8 typedef int32 TPVCmnCommandStatus

5.1.2.9 typedef int32 TPVCmnCommandType

5.1.2.10 typedef int32 TPVCmnEventType

5.1.2.11 typedef void* TPVCmnExclusivePtr

5.1.2.12 typedef void* TPVCmnInterfacePtr

5.1.2.13 typedef uint8* TPVCmnMIMEType

5.1.2.14 typedef int32 TPVCmnResponseType

5.1.2.15 typedef int32 TPVCmnSDKModuleInfo

5.1.2.16 typedef uint32 TPVCmnUUID

5.2 pv_config_interface.h File Reference

```
#include "oscl_base.h"  
#include "oscl_vector.h"
```

Data Structures

- class [PVConfigInterface](#)

5.3 pv_engine_observer.h File Reference

```
#include "pv_engine_observer_message.h"
#include "oscl_base.h"
#include "oscl_mem.h"
#include "pvmf_return_codes.h"
#include "pvmf_event_handling.h"
#include "oscl_string.h"
#include "oscl_string_containers.h"
#include "pvmf_format_type.h"
#include "pv_uuid.h"
#include "pv_interface.h"
#include "oscl_vector.h"
#include "pvmf_errorinfomessage_extension.h"
```

Data Structures

- class [PVErrorEventObserver](#)
- class [PVInformationalEventObserver](#)
- class [PVCommandStatusObserver](#)

5.4 pv_engine_observer_message.h File Reference

```
#include "oscl_base.h"
#include "oscl_mem.h"
#include "pvmf_return_codes.h"
#include "pvmf_event_handling.h"
#include "pv_engine_types.h"
#include "pvmf_errorinfomessage_extension.h"
```

Data Structures

- class [PVCmdResponse](#)
- class [PVAsyncInformationalEvent](#)
- class [PVAsyncErrorEvent](#)

5.5 pv_engine_types.h File Reference

```
#include "oscl_base.h"
#include "oscl_string.h"
#include "oscl_string_containers.h"
#include "oscl_mem.h"
#include "pvmf_format_type.h"
#include "pv_uuid.h"
#include "pv_interface.h"
#include "oscl_vector.h"
```

Data Structures

- struct [PVSDKInfo](#)
- class [PVEngineCommand](#)
- class [PVEngineAsyncEvent](#)

Typedefs

- typedef int32 [PVCommandId](#)
- typedef int32 [PVEventType](#)
- typedef OsclAny * [PVExclusivePtr](#)
- typedef int32 [PVResponseType](#)
- typedef int32 [PVLogLevelInfo](#)
- typedef Oscl_Vector< OSCL_HeapString< OsclMemAllocator >, OsclMemAllocator > [PVPMetadataList](#)
- typedef int32 [PVSDKModuleInfo](#)

5.5.1 Typedef Documentation

5.5.1.1 typedef int32 PVCommandId

5.5.1.2 typedef int32 PVEventType

5.5.1.3 typedef OsclAny* PVExclusivePtr

5.5.1.4 typedef int32 PVLogLevelInfo

5.5.1.5 typedef Oscl_Vector<OSCL_HeapString<OsclMemAllocator>, OsclMemAllocator> PVPMetadataList

5.5.1.6 typedef int32 PVResponseType

5.5.1.7 typedef int32 PVSDKModuleInfo

5.6 pv_interface_cmd_message.h File Reference

```
#include "pv_common_types.h"
#include "oscl_types.h"
#include "oscl_mem.h"
#include "oscl_string_containers.h"
#include "pv_engine_types.h"
```

Data Structures

- class [CPVCmnInterfaceCmdMessage](#)

Functions

- int32 `operator<` (const [CPVCmnInterfaceCmdMessage](#) &a, const [CPVCmnInterfaceCmdMessage](#) &b)

5.6.1 Function Documentation

5.6.1.1 int32 `operator<` (const [CPVCmnInterfaceCmdMessage](#) & a, const [CPVCmnInterfaceCmdMessage](#) & b) [`inline`]

References [CPVCmnInterfaceCmdMessage::iId](#), and [CPVCmnInterfaceCmdMessage::iPriority](#).

5.7 pvauthorenginefactory.h File Reference

Data Structures

- class [PVAuthorEngineFactory](#)

5.8 pvauthorengineinterface.h File Reference

```
#include "oscl_base.h"
#include "oscl_string.h"
#include "pv_engine_types.h"
```

Data Structures

- class [PVAuthorEngineInterface](#)

Enumerations

- enum [PVAEState](#) {
 PVAE_STATE_IDLE = 0, PVAE_STATE_OPENED, PVAE_STATE_INITIALIZED, PVAE_STATE_RECORDING,
 PVAE_STATE_PAUSED, PVAE_STATE_ERROR }
• enum [PVAEErrorEvent](#) { PVAE_ENCODE_ERROR }
• enum [PVAEInfoEvent](#) { PVAE_OUTPUT_PROGRESS }

5.8.1 Enumeration Type Documentation

5.8.1.1 enum PVAEErrorEvent

Enumeration of errors from pvAuthor Engine.

Enumerator:

PVAE_ENCODE_ERROR

5.8.1.2 enum PVAEInfoEvent

Enumeration of informational events from pvAuthor Engine.

Enumerator:

PVAE_OUTPUT_PROGRESS

5.8.1.3 enum PVAEState

An enumeration of the major states of the pvAuthor Engine.

Enumerator:

PVAE_STATE_IDLE
PVAE_STATE_OPENED
PVAE_STATE_INITIALIZED
PVAE_STATE_RECORDING
PVAE_STATE_PAUSED
PVAE_STATE_ERROR

Index

- ~CPVCmnAsyncEvent
 - CPVCmnAsyncEvent, 8
- ~CPVCmnInterfaceCmdMessage
 - CPVCmnInterfaceCmdMessage, 13
- ~CPVCmnInterfaceObserverMessage
 - CPVCmnInterfaceObserverMessage, 17
- ~MPVCmnCmdStatusObserver
 - MPVCmnCmdStatusObserver, 19
- ~MPVCmnErrorEventObserver
 - MPVCmnErrorEventObserver, 20
- ~MPVCmnInfoEventObserver
 - MPVCmnInfoEventObserver, 21
- ~PVAsyncErrorEvent
 - PVAsyncErrorEvent, 22
- ~PVAsyncInformationalEvent
 - PVAsyncInformationalEvent, 24
- ~PVAuthorEngineInterface
 - PVAuthorEngineInterface, 29
- ~PVCommandStatusObserver
 - PVCommandStatusObserver, 41
- ~PVErrorEventObserver
 - PVErrorEventObserver, 50
- ~PVInformationalEventObserver
 - PVInformationalEventObserver, 51
- AddDataSink
 - PVAuthorEngineInterface, 29
- AddDataSource
 - PVAuthorEngineInterface, 29
- AddMediaTrack
 - PVAuthorEngineInterface, 29, 30
- CancelAllCommands
 - PVAuthorEngineInterface, 31
- Close
 - PVAuthorEngineInterface, 31
- CommandCompleted
 - PVCommandStatusObserver, 41
- CommandCompletedL
 - MPVCmnCmdStatusObserver, 19
- compare
 - CPVCmnInterfaceCmdMessage, 14
 - CPVCmnInterfaceObserverMessageCompare, 18
- CPVCmnAsyncErrorEvent
 - pv_common_types.h, 56
- CPVCmnAsyncEvent, 7
 - ~CPVCmnAsyncEvent, 8
 - CPVCmnAsyncEvent, 8
 - GetEventData, 8
 - GetEventType, 8
 - GetLocalBuffer, 8
 - iEventType, 8
 - iExclusivePtr, 8
 - iLocalBuffer, 8
- CPVCmnAsyncInfoEvent
 - pv_common_types.h, 56
- CPVCmnAudioCaps
 - pv_common_types.h, 56
- CPVCmnAudioPrefs
 - pv_common_types.h, 56
- CPVCmnCmdResp, 10
 - CPVCmnCmdResp, 10
 - GetCmdId, 10
 - GetCmdStatus, 10
 - GetCmdType, 11
 - GetContext, 11
 - GetResponseData, 11
 - GetResponseDataSize, 11
 - iCmdId, 11
 - iCmdType, 11
 - iContext, 11
 - iResponseData, 11
 - iResponseDataSize, 12
 - iStatus, 12
- CPVCmnInterfaceCmdMessage, 13
 - ~CPVCmnInterfaceCmdMessage, 13
 - compare, 14
 - CPVCmnInterfaceCmdMessage, 13
 - GetCommandId, 14
 - GetContextData, 14
 - GetPriority, 14
 - GetType, 14
 - iContextData, 14
 - iId, 14
 - iPriority, 14
 - iType, 14
 - operator<, 14
 - PVInterfaceProxy, 14
 - SetId, 14

- CPVCmnInterfaceObserverMessage, 16
 - ~CPVCmnInterfaceObserverMessage, 17
 - CPVCmnInterfaceObserverMessage, 17
 - GetPriority, 17
 - GetResponseType, 17
 - iOrder, 17
 - iPriority, 17
 - iResponseType, 17
- CPVCmnInterfaceObserverMessageCompare, 18
 - compare, 18
- CPVCmnVideoCaps
 - pv_common_types.h, 56
- CPVCmnVideoPrefs
 - pv_common_types.h, 56
- CreateAuthor
 - PVAuthorEngineFactory, 26
- DeleteAuthor
 - PVAuthorEngineFactory, 26
- GetAsyncEventType
 - PVEngineAsyncEvent, 44
- GetCmdId
 - CPVCmnCmdResp, 10
 - PVCmdResponse, 39
 - PVEngineCommand, 46
- GetCmdStatus
 - CPVCmnCmdResp, 10
 - PVCmdResponse, 39
- GetCmdType
 - CPVCmnCmdResp, 11
 - PVEngineCommand, 46
- GetCommandId
 - CPVCmnInterfaceCmdMessage, 14
- GetContext
 - CPVCmnCmdResp, 11
 - PVCmdResponse, 40
 - PVEngineCommand, 46
- GetContextData
 - CPVCmnInterfaceCmdMessage, 14
- GetEventData
 - CPVCmnAsyncEvent, 8
 - PVAsyncErrorEvent, 22
 - PVAsyncInformationalEvent, 24
- GetEventType
 - CPVCmnAsyncEvent, 8
 - PVAsyncErrorEvent, 22
 - PVAsyncInformationalEvent, 24
- GetExtendedErrorInfoMessage
 - PVCmdResponse, 40
- GetLocalBuffer
 - CPVCmnAsyncEvent, 8
- GetLogLevel
 - PVAuthorEngineInterface, 31
- GetMimeType
 - PVEngineCommand, 46
- GetParam1
 - PVEngineCommand, 47
- GetParam2
 - PVEngineCommand, 47
- GetParam3
 - PVEngineCommand, 47
- GetPriority
 - CPVCmnInterfaceCmdMessage, 14
 - CPVCmnInterfaceObserverMessage, 17
- GetPVAuthorState
 - PVAuthorEngineInterface, 32
- GetResponseData
 - CPVCmnCmdResp, 11
 - PVCmdResponse, 40
- GetResponseDataSize
 - CPVCmnCmdResp, 11
 - PVCmdResponse, 40
- GetResponseType
 - CPVCmnInterfaceObserverMessage, 17
 - PVAsyncErrorEvent, 23
 - PVAsyncInformationalEvent, 25
 - PVCmdResponse, 40
- GetSDKInfo
 - PVAuthorEngineInterface, 32
- GetSDKModuleInfo
 - PVAuthorEngineInterface, 32
- GetType
 - CPVCmnInterfaceCmdMessage, 14
- GetUuid
 - PVEngineCommand, 47
- HandleErrorEvent
 - PVErrorEventObserver, 50
- HandleErrorEventL
 - MPVCmnErrorEventObserver, 20
- HandleInformationalEvent
 - PVInformationalEventObserver, 51
- HandleInformationalEventL
 - MPVCmnInfoEventObserver, 21
- iAsyncEventType
 - PVEngineAsyncEvent, 44
- iCmdId
 - CPVCmnCmdResp, 11
 - PVEngineCommand, 48
- iCmdType
 - CPVCmnCmdResp, 11
 - PVEngineCommand, 48
- iContext
 - CPVCmnCmdResp, 11
- iContextData
 - CPVCmnInterfaceCmdMessage, 14

- PVEngineCommand, 48
- iDate
 - PVSDKInfo, 52
 - TPVCmnSDKInfo, 53
- iEventType
 - CPVCmnAsyncEvent, 8
- iExclusivePtr
 - CPVCmnAsyncEvent, 8
- iId
 - CPVCmnInterfaceCmdMessage, 14
- iLabel
 - PVSDKInfo, 52
 - TPVCmnSDKInfo, 53
- iLocalBuffer
 - CPVCmnAsyncEvent, 8
- iMimeType
 - PVEngineCommand, 48
- Init
 - PVAuthorEngineInterface, 33
- iOrder
 - CPVCmnInterfaceObserverMessage, 17
- iParam1
 - PVEngineCommand, 48
- iParam2
 - PVEngineCommand, 48
- iParam3
 - PVEngineCommand, 48
- iPriority
 - CPVCmnInterfaceCmdMessage, 14
 - CPVCmnInterfaceObserverMessage, 17
- iResponseData
 - CPVCmnCmdResp, 11
- iResponseDataSize
 - CPVCmnCmdResp, 12
- iResponseType
 - CPVCmnInterfaceObserverMessage, 17
- iStatus
 - CPVCmnCmdResp, 12
- iType
 - CPVCmnInterfaceCmdMessage, 14
- iUuid
 - PVEngineCommand, 48
- MPVCmnCmdStatusObserver, 19
 - ~MPVCmnCmdStatusObserver, 19
 - CommandCompletedL, 19
- MPVCmnErrorEventObserver, 20
 - ~MPVCmnErrorEventObserver, 20
 - HandleErrorEventL, 20
- MPVCmnInfoEventObserver, 21
 - ~MPVCmnInfoEventObserver, 21
 - HandleInformationalEventL, 21
- Open
 - PVAuthorEngineInterface, 33
- operator<
 - CPVCmnInterfaceCmdMessage, 14
 - pv_interface_cmd_message.h, 61
- operator=
 - PVSDKInfo, 52
 - TPVCmnSDKInfo, 53
- Pause
 - PVAuthorEngineInterface, 33
- PV_COMMON_ASYNC_EVENT_LOCAL_-
 - BUF_SIZE
 - pv_common_types.h, 56
- pv_common_types.h, 55
 - CPVCmnAsyncErrorEvent, 56
 - CPVCmnAsyncInfoEvent, 56
 - CPVCmnAudioCaps, 56
 - CPVCmnAudioPrefs, 56
 - CPVCmnVideoCaps, 56
 - CPVCmnVideoPrefs, 56
 - PV_COMMON_ASYNC_EVENT_LOCAL_-
 - BUF_SIZE, 56
 - TPVCmnCommandId, 56
 - TPVCmnCommandStatus, 56
 - TPVCmnCommandType, 56
 - TPVCmnEventType, 56
 - TPVCmnExclusivePtr, 56
 - TPVCmnInterfacePtr, 56
 - TPVCmnMIMETYPE, 56
 - TPVCmnResponseType, 56
 - TPVCmnSDKModuleInfo, 56
 - TPVCmnUUID, 56
- pv_config_interface.h, 57
- pv_engine_observer.h, 58
- pv_engine_observer_message.h, 59
- pv_engine_types.h, 60
 - PVCommandId, 60
 - PVEventType, 60
 - PVExclusivePtr, 60
 - PVLogLevelInfo, 60
 - PVPMetadataList, 60
 - PVResponseType, 60
 - PVSDKModuleInfo, 60
- pv_interface_cmd_message.h, 61
 - operator<, 61
- PVAE_ENCODE_ERROR
 - pvauthorengineinterface.h, 63
- PVAE_OUTPUT_PROGRESS
 - pvauthorengineinterface.h, 63
- PVAE_STATE_ERROR
 - pvauthorengineinterface.h, 63
- PVAE_STATE_IDLE
 - pvauthorengineinterface.h, 63
- PVAE_STATE_INITIALIZED

- pvauthorengineinterface.h, 63
- PVAE_STATE_OPENED
 - pvauthorengineinterface.h, 63
- PVAE_STATE_PAUSED
 - pvauthorengineinterface.h, 63
- PVAE_STATE_RECORDING
 - pvauthorengineinterface.h, 63
- PVAEErrorEvent
 - pvauthorengineinterface.h, 63
- PVAEInfoEvent
 - pvauthorengineinterface.h, 63
- PVAEState
 - pvauthorengineinterface.h, 63
- PVAsyncErrorEvent, 22
 - ~PVAsyncErrorEvent, 22
 - GetEventData, 22
 - GetEventType, 22
 - GetResponseType, 23
 - PVAsyncErrorEvent, 22
- PVAsyncInformationalEvent, 24
 - ~PVAsyncInformationalEvent, 24
 - GetEventData, 24
 - GetEventType, 24
 - GetResponseType, 25
 - PVAsyncInformationalEvent, 24
- PVAuthorEngineFactory, 26
 - CreateAuthor, 26
 - DeleteAuthor, 26
- pvauthorenginefactory.h, 62
- PVAuthorEngineInterface, 28
 - ~PVAuthorEngineInterface, 29
 - AddDataSink, 29
 - AddDataSource, 29
 - AddMediaTrack, 29, 30
 - CancelAllCommands, 31
 - Close, 31
 - GetLogLevel, 31
 - GetPVAuthorState, 32
 - GetSDKInfo, 32
 - GetSDKModuleInfo, 32
 - Init, 33
 - Open, 33
 - Pause, 33
 - QueryInterface, 34
 - RemoveDataSink, 34
 - RemoveDataSource, 34
 - RemoveLogAppender, 35
 - Reset, 35
 - Resume, 35
 - SelectComposer, 36
 - SetLogAppender, 37
 - SetLogLevel, 37
 - Start, 38
 - Stop, 38
- pvauthorengineinterface.h, 63
 - PVAE_ENCODE_ERROR, 63
 - PVAE_OUTPUT_PROGRESS, 63
 - PVAE_STATE_ERROR, 63
 - PVAE_STATE_IDLE, 63
 - PVAE_STATE_INITIALIZED, 63
 - PVAE_STATE_OPENED, 63
 - PVAE_STATE_PAUSED, 63
 - PVAE_STATE_RECORDING, 63
 - PVAEErrorEvent, 63
 - PVAEInfoEvent, 63
 - PVAEState, 63
- PVCmdResponse, 39
 - GetCmdId, 39
 - GetCmdStatus, 39
 - GetContext, 40
 - GetExtendedErrorInfoMessage, 40
 - GetResponseData, 40
 - GetResponseDataSize, 40
 - GetResponseType, 40
 - PVCmdResponse, 39
- PVCommandId
 - pv_engine_types.h, 60
- PVCommandStatusObserver, 41
 - ~PVCommandStatusObserver, 41
 - CommandCompleted, 41
- PVConfigInterface, 42
- PVEngineAsyncEvent, 43
 - GetAsyncEventType, 44
 - iAsyncEventType, 44
 - PVEngineAsyncEvent, 43
- PVEngineCommand, 45
 - GetCmdId, 46
 - GetCmdType, 46
 - GetContext, 46
 - GetMimeType, 46
 - GetParam1, 47
 - GetParam2, 47
 - GetParam3, 47
 - GetUuid, 47
 - iCmdId, 48
 - iCmdType, 48
 - iContextData, 48
 - iMimeType, 48
 - iParam1, 48
 - iParam2, 48
 - iParam3, 48
 - iUuid, 48
 - PVEngineCommand, 45, 46
 - SetMimeType, 47
 - SetUuid, 48
- PVErrorEventObserver, 50
 - ~PVErrorEventObserver, 50
 - HandleErrorEvent, 50

- PVEventType
 - pv_engine_types.h, 60
- PVExclusivePtr
 - pv_engine_types.h, 60
- PVInformationalEventObserver, 51
 - ~PVInformationalEventObserver, 51
 - HandleInformationalEvent, 51
- PVInterfaceProxy
 - CPVCmnInterfaceCmdMessage, 14
- PVLogLevelInfo
 - pv_engine_types.h, 60
- PVPMetadataList
 - pv_engine_types.h, 60
- PVResponseType
 - pv_engine_types.h, 60
- PVSDKInfo, 52
 - iDate, 52
 - iLabel, 52
 - operator=, 52
 - PVSDKInfo, 52
- PVSDKModuleInfo
 - pv_engine_types.h, 60
- QueryInterface
 - PVAuthorEngineInterface, 34
- RemoveDataSink
 - PVAuthorEngineInterface, 34
- RemoveDataSource
 - PVAuthorEngineInterface, 34
- RemoveLogAppender
 - PVAuthorEngineInterface, 35
- Reset
 - PVAuthorEngineInterface, 35
- Resume
 - PVAuthorEngineInterface, 35
- SelectComposer
 - PVAuthorEngineInterface, 36
- SetId
 - CPVCmnInterfaceCmdMessage, 14
- SetLogAppender
 - PVAuthorEngineInterface, 37
- SetLogLevel
 - PVAuthorEngineInterface, 37
- SetMimeType
 - PVEngineCommand, 47
- SetUuid
 - PVEngineCommand, 48
- Start
 - PVAuthorEngineInterface, 38
- Stop
 - PVAuthorEngineInterface, 38
- TPVCmnCommandId
 - pv_common_types.h, 56
- TPVCmnCommandStatus
 - pv_common_types.h, 56
- TPVCmnCommandType
 - pv_common_types.h, 56
- TPVCmnEventType
 - pv_common_types.h, 56
- TPVCmnExclusivePtr
 - pv_common_types.h, 56
- TPVCmnInterfacePtr
 - pv_common_types.h, 56
- TPVCmnMIMEType
 - pv_common_types.h, 56
- TPVCmnResponseType
 - pv_common_types.h, 56
- TPVCmnSDKInfo, 53
 - iDate, 53
 - iLabel, 53
 - operator=, 53
 - TPVCmnSDKInfo, 53
- TPVCmnSDKModuleInfo
 - pv_common_types.h, 56
- TPVCmnUUID
 - pv_common_types.h, 56