



PV Author Engine

Build Version: CORE\_8.501.1.1

February 15, 2010

# Contents

<b>1</b>	<b><a href="#">pvauthor_engine Hierarchical Index</a></b>	<b><a href="#">1</a></b>
1.1	<a href="#">pvauthor_engine Class Hierarchy</a>	<a href="#">1</a>
<b>2</b>	<b><a href="#">pvauthor_engine Data Structure Index</a></b>	<b><a href="#">2</a></b>
2.1	<a href="#">pvauthor_engine Data Structures</a>	<a href="#">2</a>
<b>3</b>	<b><a href="#">pvauthor_engine File Index</a></b>	<b><a href="#">3</a></b>
3.1	<a href="#">pvauthor_engine File List</a>	<a href="#">3</a>
<b>4</b>	<b><a href="#">pvauthor_engine Data Structure Documentation</a></b>	<b><a href="#">4</a></b>
4.1	<a href="#">CPVCmnAsyncEvent Class Reference</a>	<a href="#">4</a>
4.2	<a href="#">CPVCmnCmdResp Class Reference</a>	<a href="#">6</a>
4.3	<a href="#">CPVCmnInterfaceCmdMessage Class Reference</a>	<a href="#">8</a>
4.4	<a href="#">CPVCmnInterfaceObserverMessage Class Reference</a>	<a href="#">10</a>
4.5	<a href="#">CPVCmnInterfaceObserverMessageCompare Class Reference</a>	<a href="#">12</a>
4.6	<a href="#">MPVCmnCmdStatusObserver Class Reference</a>	<a href="#">13</a>
4.7	<a href="#">MPVCmnErrorEventObserver Class Reference</a>	<a href="#">14</a>
4.8	<a href="#">MPVCmnInfoEventObserver Class Reference</a>	<a href="#">15</a>
4.9	<a href="#">PVAsyncErrorEvent Class Reference</a>	<a href="#">16</a>
4.10	<a href="#">PVAsyncInformationalEvent Class Reference</a>	<a href="#">18</a>
4.11	<a href="#">PVAuthorEngineFactory Class Reference</a>	<a href="#">20</a>
4.12	<a href="#">PVAuthorEngineInterface Class Reference</a>	<a href="#">22</a>
4.13	<a href="#">PVCmdResponse Class Reference</a>	<a href="#">33</a>
4.14	<a href="#">PVCommandStatusObserver Class Reference</a>	<a href="#">35</a>
4.15	<a href="#">PVConfigInterface Class Reference</a>	<a href="#">36</a>
4.16	<a href="#">PVEngineAsyncEvent Class Reference</a>	<a href="#">37</a>
4.17	<a href="#">PVEngineCommand Class Reference</a>	<a href="#">39</a>
4.18	<a href="#">PVErrorEventObserver Class Reference</a>	<a href="#">43</a>
4.19	<a href="#">PVInformationalEventObserver Class Reference</a>	<a href="#">44</a>

4.20	PVSDKInfo Struct Reference . . . . .	45
4.21	TPVCmnSDKInfo Struct Reference . . . . .	46
<b>5</b>	<b>pvauthor_engine File Documentation</b>	<b>47</b>
5.1	pv_common_types.h File Reference . . . . .	47
5.2	pv_config_interface.h File Reference . . . . .	49
5.3	pv_engine_observer.h File Reference . . . . .	50
5.4	pv_engine_observer_message.h File Reference . . . . .	51
5.5	pv_engine_types.h File Reference . . . . .	52
5.6	pv_interface_cmd_message.h File Reference . . . . .	53
5.7	pvauthorenginefactory.h File Reference . . . . .	54
5.8	pvauthorengineinterface.h File Reference . . . . .	55

# Chapter 1

## pvauthor\_engine Hierarchical Index

### 1.1 pvauthor\_engine Class Hierarchy

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

CPVCmnInterfaceCmdMessage . . . . .	8
CPVCmnInterfaceObserverMessage . . . . .	10
CPVCmnAsyncEvent . . . . .	4
CPVCmnCmdResp . . . . .	6
CPVCmnInterfaceObserverMessageCompare . . . . .	12
MPVCmnCmdStatusObserver . . . . .	13
MPVCmnErrorEventObserver . . . . .	14
MPVCmnInfoEventObserver . . . . .	15
PVAsyncErrorEvent . . . . .	16
PVAsyncInformationalEvent . . . . .	18
PVAuthorEngineFactory . . . . .	20
PVAuthorEngineInterface . . . . .	22
PVCmdResponse . . . . .	33
PVCommandStatusObserver . . . . .	35
PVConfigInterface . . . . .	36
PVEngineAsyncEvent . . . . .	37
PVEngineCommand . . . . .	39
PVErrorEventObserver . . . . .	43
PVInformationalEventObserver . . . . .	44
PVSDKInfo . . . . .	45
TPVCmnSDKInfo . . . . .	46

## Chapter 2

# pvauthor\_engine Data Structure Index

### 2.1 pvauthor\_engine Data Structures

Here are the data structures with brief descriptions:

CPVCmnAsyncEvent	4
CPVCmnCmdResp	6
CPVCmnInterfaceCmdMessage	8
CPVCmnInterfaceObserverMessage	10
CPVCmnInterfaceObserverMessageCompare	12
MPVCmnCmdStatusObserver	13
MPVCmnErrorEventObserver	14
MPVCmnInfoEventObserver	15
PVAsyncErrorEvent	16
PVAsyncInformationalEvent	18
PVAuthorEngineFactory	20
PVAuthorEngineInterface	22
PVCmdResponse	33
PVCommandStatusObserver	35
PVConfigInterface	36
PVEngineAsyncEvent	37
PVEngineCommand	39
PVErrorEventObserver	43
PVInformationalEventObserver	44
PVSDKInfo	45
TPVCmnSDKInfo	46

## Chapter 3

# pvauthor\_engine File Index

### 3.1 pvauthor\_engine File List

Here is a list of all files with brief descriptions:

<a href="#">pv_common_types.h</a>	47
<a href="#">pv_config_interface.h</a>	49
<a href="#">pv_engine_observer.h</a>	50
<a href="#">pv_engine_observer_message.h</a>	51
<a href="#">pv_engine_types.h</a>	52
<a href="#">pv_interface_cmd_message.h</a>	53
<a href="#">pvauthorenginefactory.h</a>	54
<a href="#">pvauthorengineinterface.h</a>	55

## Chapter 4

# pvaauthor\_engine Data Structure Documentation

### 4.1 CPVCmnAsyncEvent Class Reference

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

Inheritance diagram for CPVCmnAsyncEvent::



#### Public Methods

- [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]

**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.

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

### Returns:

Returns the Event type that has been received

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

### Returns:

Returns the local data associated with the event.

## 4.1.4 Field Documentation

**4.1.4.1** [TPVCmnEventType](#) CPVCmnAsyncEvent::iEventType [protected]

**4.1.4.2** [TPVCmnExclusivePtr](#) CPVCmnAsyncEvent::iExclusivePtr [protected]

**4.1.4.3** uint8 CPVCmnAsyncEvent::iLocalBuffer[PV\_COMMON\_ASYNC\_EVENT\_LOCAL\_BUF\_SIZE] [protected]

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 Methods

- [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.

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

Returns the completion status of the command

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

Returns the command type that is being completed.

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

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

**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

**4.2.2.6 int** CPVCmnCmdResp::GetResponseDataSize () const [inline]**4.2.3 Field Documentation****4.2.3.1 TPVCmnCommandId** CPVCmnCmdResp::iCmdId [protected]**4.2.3.2 TPVCmnCommandType** CPVCmnCmdResp::iCmdType [protected]**4.2.3.3 void\*** CPVCmnCmdResp::iContext [protected]**4.2.3.4 void\*** CPVCmnCmdResp::iResponseData [protected]**4.2.3.5 int** CPVCmnCmdResp::iResponseDataSize [protected]**4.2.3.6 TPVCmnCommandStatus** CPVCmnCmdResp::iStatus [protected]

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 Methods

- [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*, OsciAny \* *aContextData*) [inline]

**4.3.2.2** CPVcmnInterfaceCmdMessage::CPVcmnInterfaceCmdMessage () [inline]

**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

**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]

**4.3.3.5** int CPVcmnInterfaceCmdMessage::GetType () [inline]

**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]

**4.3.5.2** [PVCommandId](#) CPVcmnInterfaceCmdMessage::iId [protected]

**4.3.5.3** int32 CPVcmnInterfaceCmdMessage::iPriority [protected]

**4.3.5.4** int CPVcmnInterfaceCmdMessage::iType [protected]

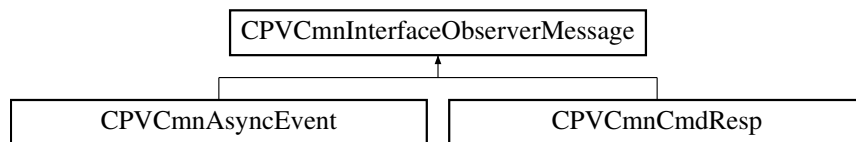
The documentation for this class was generated from the following file:

- [pv\\_interface\\_cmd\\_message.h](#)

## 4.4 CPVCmnInterfaceObserverMessage Class Reference

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

Inheritance diagram for CPVCmnInterfaceObserverMessage::



### Public Methods

- [CPVCmnInterfaceObserverMessage \(\)](#)
- [CPVCmnInterfaceObserverMessage \(TPVCmnResponseType aResponseType\)](#)
- [virtual ~CPVCmnInterfaceObserverMessage \(\)](#)
- [TPVCmnResponseType GetResponseType \(\) const](#)
- [virtual int GetPriority \(\) const](#)

### Data Fields

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

### 4.4.1 Detailed Description

CPVCmnInterfaceObserverMessage Class

CPVCmnInterfaceObserverMessage 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]

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

## 4.4.4 Field Documentation

4.4.4.1 int CPVCmnInterfaceObserverMessage::iOrder

4.4.4.2 int CPVCmnInterfaceObserverMessage::iPriority

4.4.4.3 TPVCmnResponseType CPVCmnInterfaceObserverMessage::iResponseType

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 Methods

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

### 4.5.1 Member Function Documentation

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

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 Methods

- 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 Methods

- 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 Methods

- 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 PVAAsyncErrorEvent Class Reference

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

### Public Methods

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

### 4.9.1 Detailed Description

PVAAsyncErrorEvent Class

PVAAsyncErrorEvent 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** **PVAAsyncErrorEvent::PVAAsyncErrorEvent** (**PVEventType** aEventType, **PVExclusivePtr** aEventData = NULL, uint8 \* aLocalBuffer = NULL, int32 aLocalBufferSize = 0) [inline]

Constructor for PVAAsyncErrorEvent

**4.9.2.2** **PVAAsyncErrorEvent::PVAAsyncErrorEvent** (**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** **PVAAsyncErrorEvent::~~PVAAsyncErrorEvent** () [inline]

Destructor

### 4.9.3 Member Function Documentation

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

**Returns:**

Returns the opaque data associated with the event.

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

Returns the Event type that has been received

**4.9.3.3 PVResponseType PVAAsyncErrorEvent::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 PVAAsyncInformationalEvent Class Reference

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

### Public Methods

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

### 4.10.1 Detailed Description

PVAAsyncInformationalEvent Class

PVAAsyncInformationalEvent 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 PVAAsyncInformationalEvent::PVAAsyncInformationalEvent** (**PVEventType** aEventType, **PVExclusivePtr** aEventData = NULL, uint8 \*aLocalBuffer = NULL, int32 aLocalBufferSize = 0) [inline]

Constructor for PVAAsyncInformationalEvent

**4.10.2.2 PVAAsyncInformationalEvent::PVAAsyncInformationalEvent** (**PVEventType** aEventType, OsclAny \*aContext, PVInterface \*aEventExtInterface, **PVExclusivePtr** aEventData = NULL, uint8 \*aLocalBuffer = NULL, int32 aLocalBufferSize = 0) [inline]

Constructor with context and event extension interface

**4.10.2.3 PVAAsyncInformationalEvent::~~PVAAsyncInformationalEvent** () [inline]

Destructor

### 4.10.3 Member Function Documentation

**4.10.3.1 void PVAAsyncInformationalEvent::GetEventData** (**PVExclusivePtr** &aPtr) const [inline]

#### Returns:

Returns the opaque data associated with the event.

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

Returns the Event type that has been received

**4.10.3.3 PVResponseType PVAAsyncInformationalEvent::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 Methods

- OSCL\_IMPORT\_REF [PVAuthorEngineInterface](#) \* [CreateAuthor](#) ([PVCommandStatusObserver](#) \*aCmdStatusObserver, [PVErrrorEventObserver](#) \*aErrorEventObserver, [PVInformationalEventObserver](#) \*aInfoEventObserver)
- 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** OSCL\_IMPORT\_REF [PVAuthorEngineInterface](#)\* [PVAuthorEngineFactory::CreateAuthor](#) ([PVCommandStatusObserver](#) \*aCmdStatusObserver, [PVErrrorEventObserver](#) \*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** 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 Methods

- 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 PVMFNodeInterface &aDataSource, const OsciAny \*aContextData=NULL)=0
- virtual [PVCommandId RemoveDataSource](#) (const PVMFNodeInterface &aDataSource, const OsciAny \*aContextData=NULL)=0
- virtual [PVCommandId SelectComposer](#) (const PvmfMimeString &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 PVMFNodeInterface &aDataSource, const PvmfMimeString &aEncoderType, const OsciAny \*aComposer, PVInterface \*&aConfigInterface, const OsciAny \*aContextData=NULL)=0
- virtual [PVCommandId AddMediaTrack](#) (const PVMFNodeInterface &aDataSource, const PVUuid &aEncoderUuid, const OsciAny \*aComposer, PVInterface \*&aConfigInterface, const OsciAny \*aContextData=NULL)=0
- virtual [PVCommandId AddDataSink](#) (const PVMFNodeInterface &aDataSink, const OsciAny \*aComposer, const OsciAny \*aContextData=NULL)=0
- virtual [PVCommandId RemoveDataSink](#) (const PVMFNodeInterface &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 QueryUUID](#) (const PvmfMimeString &aMimeType, Osci\_Vector< PVUuid, OsciMemAllocator > &aUuids, bool aExactUuidsOnly=false, const OsciAny \*aContextData=NULL)=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 Methods

- 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 OsciAny \* *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 OsciAny \* *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 OsciAny \* *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 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 **OscAny** \* ***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 UUID and description string for each module. The PV UUID 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 UIID, 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::QueryUUID (const PvmfMimeString & *aMimeType*, Oscl\_Vector< PVUuid, OsclMemAllocator > & *aUuids*, bool *aExactUuidsOnly* = false, const OsclAny \* *aContextData* = NULL) [pure virtual]

Discover the UUIDs of interfaces associated with the specified MIME type and node

This API is to allow for extensibility of the pvAuthor Engine interface. User can query for all UUIDs associated with a particular MIME type. The UUIDs will be added to the aUuids vector provided by the user. Currently this API is NOT SUPPORTED.

**Parameters:**

*aMimeType* The MIME type of the desired interfaces  
*aUuids* A vector to hold the discovered UUIDs  
*aExactUuidsOnly* Turns on/off the retrieval of UUIDs with aMimeType as a base type  
*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::RemoveDataSink (const PVMFNodeInterface & *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

**4.12.3.17** virtual **PVCommandId** PVAuthorEngineInterface::RemoveDataSource (const PVMFNodeInterface & *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

**4.12.3.18** 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

**4.12.3.19** 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

**4.12.3.20** **virtual PVCommandId PVAuthorEngineInterface::Resume** (const OsciAny \* *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

**4.12.3.21** **virtual PVCommandId PVAuthorEngineInterface::SelectComposer** (const PVUuid & *aComposerUuid*, PVInterface \*& *aConfigInterface*, const OsciAny \* *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 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:

***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

**4.12.3.22** 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

**4.12.3.23** 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

**4.12.3.24** 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

#### 4.12.3.25 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

#### 4.12.3.26 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 Methods

- [PVCmdResponse](#) ([PVCommandId](#) aId, [OsclAny](#) \*aContext, [PVMFStatus](#) aStatus, [OsclAny](#) \*aEventData=NULL, [int32](#) aEventDataSize=0)
- [PVCmdResponse](#) ([PVCommandId](#) aId, [OsclAny](#) \*aContext, [PVMFStatus](#) aStatus, [PVInterface](#) \*aEventExtInterface=NULL, [OsclAny](#) \*aEventData=NULL, [int32](#) aEventDataSize=0)
- [PVResponseType](#) [GetResponseType](#) () const
- [PVCommandId](#) [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](#) ([PVCommandId](#) aId, [OsclAny](#) \* aContext, [PVMFStatus](#) aStatus, [OsclAny](#) \* aEventData = NULL, [int32](#) aEventDataSize = 0) [inline]

Constructor for PVCmdResponse

**4.13.2.2** [PVCmdResponse::PVCmdResponse](#) ([PVCommandId](#) 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** [PVCommandId](#) [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** `OsclAny* 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** `OsclAny* PVCmdResponse::GetResponseData () const` [inline]

WILL BE DEPRECATED WHEN PVMFCmdResp REMOVES EVENT DATA

**Returns:**

Returns additional data asociated 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 Methods

- 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 Methods

- [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 Osci\_Vector.

##### Parameters:

- aAsyncEvent* The reference to the source PVEngineAsyncEvent to copy the data values from.

##### Returns:

None



### 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.

### 4.16.4 Field Documentation

#### 4.16.4.1 int32 PVEngineAsyncEvent::iAsyncEventType

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 Methods

- [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

## 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.

**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.

**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

**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

**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

**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

**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

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

This function returns Uuid parameter for this command

**Returns:**

The Uuid parameter for this command

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

This function stores Mime type parameter of this command

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

This function stores the Uuid parameter of this command

#### 4.17.4 Field Documentation

4.17.4.1 [PVCommandId](#) PVEngineCommand::iCmdId

4.17.4.2 int32 PVEngineCommand::iCmdType

4.17.4.3 OsclAny\* PVEngineCommand::iContextData

4.17.4.4 OSCL\_HeapString<OsclMemAllocator> PVEngineCommand::iMimeType

4.17.4.5 OsclAny\* PVEngineCommand::iParam1

4.17.4.6 OsclAny\* PVEngineCommand::iParam2

4.17.4.7 OsclAny\* PVEngineCommand::iParam3

4.17.4.8 PVUuid PVEngineCommand::iUuid

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

- [pv\\_engine\\_types.h](#)

## 4.18 PVErrEventObserver Class Reference

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

### Public Methods

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

### 4.18.1 Detailed Description

PVErrEventObserver Class

PVErrEventObserver 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 PVErrEventObserver 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 PVErrEventObserver::~~PVErrEventObserver () [inline, virtual]

### 4.18.3 Member Function Documentation

**4.18.3.1** virtual void PVErrEventObserver::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 Methods

- 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 Methods

- [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](#)]

### 4.20.2 Member Function Documentation

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

### 4.20.3 Field Documentation

**4.20.3.1** [uint32 PVSDKInfo::iDate](#)

**4.20.3.2** [OSCL\\_StackString<80> PVSDKInfo::iLabel](#)

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 Methods

- [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](#)]

### 4.21.2 Member Function Documentation

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

### 4.21.3 Field Documentation

**4.21.3.1** [uint32 TPVCmnSDKInfo::iDate](#)

**4.21.3.2** [OSCL\\_StackString<80> TPVCmnSDKInfo::iLabel](#)

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

- [pv\\_common\\_types.h](#)

## Chapter 5

# pvauthor\_engine 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

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

#### 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`

### 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"
```

### Data Structures

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

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

## 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

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

### Typedefs

- typedef int32 [PVCommandId](#)
- typedef int32 [PVEventType](#)
- typedef OsciAny \* [PVExclusivePtr](#)
- typedef int32 [PVResponseType](#)
- typedef int32 [PVLogLevelInfo](#)
- typedef Osci\_Vector< OSCI\_HeapString< OsciMemAllocator >, OsciMemAllocator > [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 OsciAny\* PVExclusivePtr

##### 5.5.1.4 typedef int32 PVLogLevelInfo

##### 5.5.1.5 typedef Osci\_Vector<OSCL\_HeapString<OsciMemAllocator>, OsciMemAllocator> 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 "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]



## 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.

**Enumeration values:**

**PVAE\_ENCODE\_ERROR**

#### 5.8.1.2 enum PVAEInfoEvent

Enumeration of informational events from pvAuthor Engine.

**Enumeration values:**

**PVAE\_OUTPUT\_PROGRESS**

#### 5.8.1.3 enum PVAEState

An enumeration of the major states of the pvAuthor Engine.

**Enumeration values:**

**PVAE\_STATE\_IDLE**

**PVAE\_STATE\_OPENED**

**PVAE\_STATE\_INITIALIZED**

**PVAE\_STATE\_RECORDING**

**PVAE\_STATE\_PAUSED**

**PVAE\_STATE\_ERROR**

# Index

- ~CPVCmnAsyncEvent
  - CPVCmnAsyncEvent, 5
- ~CPVCmnInterfaceCmdMessage
  - CPVCmnInterfaceCmdMessage, 9
- ~CPVCmnInterfaceObserverMessage
  - CPVCmnInterfaceObserverMessage, 11
- ~MPVCmnCmdStatusObserver
  - MPVCmnCmdStatusObserver, 13
- ~MPVCmnErrorEventObserver
  - MPVCmnErrorEventObserver, 14
- ~MPVCmnInfoEventObserver
  - MPVCmnInfoEventObserver, 15
- ~PVAsyncErrorEvent
  - PVAsyncErrorEvent, 16
- ~PVAsyncInformationalEvent
  - PVAsyncInformationalEvent, 18
- ~PVAuthorEngineInterface
  - PVAuthorEngineInterface, 23
- ~PVCommandStatusObserver
  - PVCommandStatusObserver, 35
- ~PVErrorEventObserver
  - PVErrorEventObserver, 43
- ~PVInformationalEventObserver
  - PVInformationalEventObserver, 44
- AddDataSink
  - PVAuthorEngineInterface, 23
- AddDataSource
  - PVAuthorEngineInterface, 23
- AddMediaTrack
  - PVAuthorEngineInterface, 23, 24
- CancelAllCommands
  - PVAuthorEngineInterface, 25
- Close
  - PVAuthorEngineInterface, 25
- CommandCompleted
  - PVCommandStatusObserver, 35
- CommandCompletedL
  - MPVCmnCmdStatusObserver, 13
- compare
  - CPVCmnInterfaceCmdMessage, 9
  - CPVCmnInterfaceObserverMessage-Compare, 12
- CPVCmnAsyncErrorEvent
  - pv\_common\_types.h, 48
- CPVCmnAsyncEvent, 4
  - CPVCmnAsyncEvent, 5
- CPVCmnAsyncEvent
  - ~CPVCmnAsyncEvent, 5
  - CPVCmnAsyncEvent, 5
  - GetEventData, 5
  - GetEventType, 5
  - GetLocalBuffer, 5
  - iEventType, 5
  - iExclusivePtr, 5
  - iLocalBuffer, 5
- CPVCmnAsyncInfoEvent
  - pv\_common\_types.h, 48
- CPVCmnAudioCaps
  - pv\_common\_types.h, 48
- CPVCmnAudioPrefs
  - pv\_common\_types.h, 48
- CPVCmnCmdResp, 6
  - CPVCmnCmdResp, 6
- CPVCmnCmdResp
  - CPVCmnCmdResp, 6
  - GetCmdId, 6
  - GetCmdStatus, 6
  - GetCmdType, 7
  - GetContext, 7
  - GetResponseData, 7
  - GetResponseDataSize, 7
  - iCmdId, 7
  - iCmdType, 7
  - iContext, 7
  - iResponseData, 7
  - iResponseDataSize, 7
  - iStatus, 7
- CPVCmnInterfaceCmdMessage, 8
  - CPVCmnInterfaceCmdMessage, 9
- CPVCmnInterfaceCmdMessage
  - ~CPVCmnInterfaceCmdMessage, 9
  - compare, 9
  - CPVCmnInterfaceCmdMessage, 9
  - GetCommandId, 9
  - GetContextData, 9
  - GetPriority, 9
  - GetType, 9
  - iContextData, 9

- iId, 9
- iPriority, 9
- iType, 9
- operator<, 9
- PVInterfaceProxy, 9
- SetId, 9
- CPVCmnInterfaceObserverMessage, 10
  - CPVCmnInterfaceObserverMessage, 11
- CPVCmnInterfaceObserverMessage
  - ~CPVCmnInterfaceObserverMessage, 11
  - CPVCmnInterfaceObserverMessage, 11
  - GetPriority, 11
  - GetResponseType, 11
  - iOrder, 11
  - iPriority, 11
  - iResponseType, 11
- CPVCmnInterfaceObserverMessageCompare, 12
  - compare, 12
- CPVCmnVideoCaps
  - pv\_common\_types.h, 48
- CPVCmnVideoPrefs
  - pv\_common\_types.h, 48
- CreateAuthor
  - PVAuthorEngineFactory, 20
- DeleteAuthor
  - PVAuthorEngineFactory, 20
- GetAsyncEventType
  - PVEngineAsyncEvent, 38
- GetCmdId
  - CPVCmnCmdResp, 6
  - PVCmdResponse, 33
  - PVEngineCommand, 40
- GetCmdStatus
  - CPVCmnCmdResp, 6
  - PVCmdResponse, 33
- GetCmdType
  - CPVCmnCmdResp, 7
  - PVEngineCommand, 40
- GetCommandId
  - CPVCmnInterfaceCmdMessage, 9
- GetContext
  - CPVCmnCmdResp, 7
  - PVCmdResponse, 33
  - PVEngineCommand, 40
- GetContextData
  - CPVCmnInterfaceCmdMessage, 9
- GetEventData
  - CPVCmnAsyncEvent, 5
  - PVAsyncErrorEvent, 16
  - PVAsyncInformationalEvent, 18
- GetEventType
  - CPVCmnAsyncEvent, 5
  - PVAsyncErrorEvent, 16
  - PVAsyncInformationalEvent, 18
- GetExtendedErrorInfoMessage
  - PVCmdResponse, 34
- GetLocalBuffer
  - CPVCmnAsyncEvent, 5
- GetLogLevel
  - PVAuthorEngineInterface, 25
- GetMimeType
  - PVEngineCommand, 40
- GetParam1
  - PVEngineCommand, 40
- GetParam2
  - PVEngineCommand, 41
- GetParam3
  - PVEngineCommand, 41
- GetPriority
  - CPVCmnInterfaceCmdMessage, 9
  - CPVCmnInterfaceObserverMessage, 11
- GetPVAuthorState
  - PVAuthorEngineInterface, 26
- GetResponseData
  - CPVCmnCmdResp, 7
  - PVCmdResponse, 34
- GetResponseDataSize
  - CPVCmnCmdResp, 7
  - PVCmdResponse, 34
- GetResponseType
  - CPVCmnInterfaceObserverMessage, 11
  - PVAsyncErrorEvent, 17
  - PVAsyncInformationalEvent, 19
  - PVCmdResponse, 34
- GetSDKInfo
  - PVAuthorEngineInterface, 26
- GetSDKModuleInfo
  - PVAuthorEngineInterface, 26
- GetType
  - CPVCmnInterfaceCmdMessage, 9
- GetUuid
  - PVEngineCommand, 41
- HandleErrorEvent
  - PVErrorEventObserver, 43
- HandleErrorEventL
  - MPVCmnErrorEventObserver, 14
- HandleInformationalEvent
  - PVInformationalEventObserver, 44
- HandleInformationalEventL
  - MPVCmnInfoEventObserver, 15
- iAsyncEventType
  - PVEngineAsyncEvent, 38

- iCmdId
  - CPVCmnCmdResp, 7
  - PVEngineCommand, 42
- iCmdType
  - CPVCmnCmdResp, 7
  - PVEngineCommand, 42
- iContext
  - CPVCmnCmdResp, 7
- iContextData
  - CPVCmnInterfaceCmdMessage, 9
  - PVEngineCommand, 42
- iDate
  - PVSDKInfo, 45
  - TPVCmnSDKInfo, 46
- iEventType
  - CPVCmnAsyncEvent, 5
- iExclusivePtr
  - CPVCmnAsyncEvent, 5
- iId
  - CPVCmnInterfaceCmdMessage, 9
- iLabel
  - PVSDKInfo, 45
  - TPVCmnSDKInfo, 46
- iLocalBuffer
  - CPVCmnAsyncEvent, 5
- iMimeType
  - PVEngineCommand, 42
- Init
  - PVAuthorEngineInterface, 26
- iOrder
  - CPVCmnInterfaceObserverMessage, 11
- iParam1
  - PVEngineCommand, 42
- iParam2
  - PVEngineCommand, 42
- iParam3
  - PVEngineCommand, 42
- iPriority
  - CPVCmnInterfaceCmdMessage, 9
  - CPVCmnInterfaceObserverMessage, 11
- iResponseData
  - CPVCmnCmdResp, 7
- iResponseDataSize
  - CPVCmnCmdResp, 7
- iResponseType
  - CPVCmnInterfaceObserverMessage, 11
- iStatus
  - CPVCmnCmdResp, 7
- iType
  - CPVCmnInterfaceCmdMessage, 9
- iUuid
  - PVEngineCommand, 42
- MPVCmnCmdStatusObserver
  - ~MPVCmnCmdStatusObserver, 13
  - CommandCompletedL, 13
- MPVCmnErrorEventObserver, 14
- MPVCmnErrorEventObserver
  - ~MPVCmnErrorEventObserver, 14
  - HandleErrorEventL, 14
- MPVCmnInfoEventObserver, 15
- MPVCmnInfoEventObserver
  - ~MPVCmnInfoEventObserver, 15
  - HandleInformationalEventL, 15
- Open
  - PVAuthorEngineInterface, 27
- operator<
  - CPVCmnInterfaceCmdMessage, 9
  - pv\_interface\_cmd\_message.h, 53
- operator=
  - PVSDKInfo, 45
  - TPVCmnSDKInfo, 46
- Pause
  - PVAuthorEngineInterface, 27
- PV\_COMMON\_ASYNC\_EVENT\_LOCAL\_
  - BUF\_SIZE
  - pv\_common\_types.h, 48
- pv\_common\_types.h, 47
  - CPVCmnAsyncErrorEvent, 48
  - CPVCmnAsyncInfoEvent, 48
  - CPVCmnAudioCaps, 48
  - CPVCmnAudioPrefs, 48
  - CPVCmnVideoCaps, 48
  - CPVCmnVideoPrefs, 48
  - PV\_COMMON\_ASYNC\_EVENT\_
    - LOCAL\_BUF\_SIZE, 48
  - TPVCmnCommandId, 48
  - TPVCmnCommandStatus, 48
  - TPVCmnCommandType, 48
  - TPVCmnEventType, 48
  - TPVCmnExclusivePtr, 48
  - TPVCmnInterfacePtr, 48
  - TPVCmnMIMEType, 48
  - TPVCmnResponseType, 48
  - TPVCmnSDKModuleInfo, 48
  - TPVCmnUUID, 48
- pv\_config\_interface.h, 49
- pv\_engine\_observer.h, 50
- pv\_engine\_observer\_message.h, 51
- pv\_engine\_types.h, 52
  - PVCommandId, 52
  - PVEventType, 52
  - PVExclusivePtr, 52
  - PVLogLevelInfo, 52
  - PVPMetadataList, 52

- PVResponseType, 52
- PVSDKModuleInfo, 52
- pv\_interface\_cmd\_message.h, 53
  - operator<, 53
- PVAE\_ENCODE\_ERROR
  - pvauthorengineinterface.h, 55
- PVAE\_OUTPUT\_PROGRESS
  - pvauthorengineinterface.h, 55
- PVAE\_STATE\_ERROR
  - pvauthorengineinterface.h, 55
- PVAE\_STATE\_IDLE
  - pvauthorengineinterface.h, 55
- PVAE\_STATE\_INITIALIZED
  - pvauthorengineinterface.h, 55
- PVAE\_STATE\_OPENED
  - pvauthorengineinterface.h, 55
- PVAE\_STATE\_PAUSED
  - pvauthorengineinterface.h, 55
- PVAE\_STATE\_RECORDING
  - pvauthorengineinterface.h, 55
- PVAEErrorEvent
  - pvauthorengineinterface.h, 55
- PVAEInfoEvent
  - pvauthorengineinterface.h, 55
- PVAEState
  - pvauthorengineinterface.h, 55
- PVAsyncErrorEvent, 16
  - PVAsyncErrorEvent, 16
- PVAsyncErrorEvent
  - ~PVAsyncErrorEvent, 16
  - GetEventData, 16
  - GetEventType, 16
  - GetResponseType, 17
  - PVAsyncErrorEvent, 16
- PVAsyncInformationalEvent, 18
  - PVAsyncInformationalEvent, 18
- PVAsyncInformationalEvent
  - ~PVAsyncInformationalEvent, 18
  - GetEventData, 18
  - GetEventType, 18
  - GetResponseType, 19
  - PVAsyncInformationalEvent, 18
- PVAuthorEngineFactory, 20
  - CreateAuthor, 20
  - DeleteAuthor, 20
- pvauthorenginefactory.h, 54
- PVAuthorEngineInterface, 22
- PVAuthorEngineInterface
  - ~PVAuthorEngineInterface, 23
  - AddDataSink, 23
  - AddDataSource, 23
  - AddMediaTrack, 23, 24
  - CancelAllCommands, 25
  - Close, 25
  - GetLogLevel, 25
  - GetPVAuthorState, 26
  - GetSDKInfo, 26
  - GetSDKModuleInfo, 26
  - Init, 26
  - Open, 27
  - Pause, 27
  - QueryInterface, 27
  - QueryUUID, 28
  - RemoveDataSink, 28
  - RemoveDataSource, 28
  - RemoveLogAppender, 29
  - Reset, 29
  - Resume, 30
  - SelectComposer, 30
  - SetLogAppender, 31
  - SetLogLevel, 31
  - Start, 32
  - Stop, 32
- pvauthorengineinterface.h, 55
  - PVAE\_ENCODE\_ERROR, 55
  - PVAE\_OUTPUT\_PROGRESS, 55
  - PVAE\_STATE\_ERROR, 55
  - PVAE\_STATE\_IDLE, 55
  - PVAE\_STATE\_INITIALIZED, 55
  - PVAE\_STATE\_OPENED, 55
  - PVAE\_STATE\_PAUSED, 55
  - PVAE\_STATE\_RECORDING, 55
  - PVAEErrorEvent, 55
  - PVAEInfoEvent, 55
  - PVAEState, 55
- PVCmdResponse, 33
  - PVCmdResponse, 33
- PVCmdResponse
  - GetCmdId, 33
  - GetCmdStatus, 33
  - GetContext, 33
  - GetExtendedErrorInfoMessage, 34
  - GetResponseData, 34
  - GetResponseDataSize, 34
  - GetResponseType, 34
  - PVCmdResponse, 33
- PVCommandId
  - pv\_engine\_types.h, 52
- PVCommandStatusObserver, 35
- PVCommandStatusObserver
  - ~PVCommandStatusObserver, 35
  - CommandCompleted, 35
- PVConfigInterface, 36
- PVEngineAsyncEvent, 37
  - PVEngineAsyncEvent, 37
- PVEngineAsyncEvent
  - GetAsyncEventType, 38

- iAsyncEventType, 38
- PVEngineAsyncEvent, 37
- PVEngineCommand, 39
  - PVEngineCommand, 39, 40
- PVEngineCommand
  - GetCmdId, 40
  - GetCmdType, 40
  - GetContext, 40
  - GetMimeType, 40
  - GetParam1, 40
  - GetParam2, 41
  - GetParam3, 41
  - GetUuid, 41
  - iCmdId, 42
  - iCmdType, 42
  - iContextData, 42
  - iMimeType, 42
  - iParam1, 42
  - iParam2, 42
  - iParam3, 42
  - iUuid, 42
  - PVEngineCommand, 39, 40
  - SetMimeType, 41
  - SetUuid, 41
- PVErrorEventObserver, 43
- PVErrorEventObserver
  - ~PVErrorEventObserver, 43
  - HandleErrorEvent, 43
- PVEventType
  - pv\_engine\_types.h, 52
- PVExclusivePtr
  - pv\_engine\_types.h, 52
- PVInformationalEventObserver, 44
- PVInformationalEventObserver
  - ~PVInformationalEventObserver, 44
  - HandleInformationalEvent, 44
- PVInterfaceProxy
  - CPVCmnInterfaceCmdMessage, 9
- PVLogLevelInfo
  - pv\_engine\_types.h, 52
- PVPMetadataList
  - pv\_engine\_types.h, 52
- PVResponseType
  - pv\_engine\_types.h, 52
- PVSDKInfo, 45
  - iDate, 45
  - iLabel, 45
  - operator=, 45
  - PVSDKInfo, 45
- PVSDKModuleInfo
  - pv\_engine\_types.h, 52
- QueryInterface
  - PVAuthorEngineInterface, 27
- QueryUUID
  - PVAuthorEngineInterface, 28
- RemoveDataSink
  - PVAuthorEngineInterface, 28
- RemoveDataSource
  - PVAuthorEngineInterface, 28
- RemoveLogAppender
  - PVAuthorEngineInterface, 29
- Reset
  - PVAuthorEngineInterface, 29
- Resume
  - PVAuthorEngineInterface, 30
- SelectComposer
  - PVAuthorEngineInterface, 30
- SetId
  - CPVCmnInterfaceCmdMessage, 9
- SetLogAppender
  - PVAuthorEngineInterface, 31
- SetLogLevel
  - PVAuthorEngineInterface, 31
- SetMimeType
  - PVEngineCommand, 41
- SetUuid
  - PVEngineCommand, 41
- Start
  - PVAuthorEngineInterface, 32
- Stop
  - PVAuthorEngineInterface, 32
- TPVCmnCommandId
  - pv\_common\_types.h, 48
- TPVCmnCommandStatus
  - pv\_common\_types.h, 48
- TPVCmnCommandType
  - pv\_common\_types.h, 48
- TPVCmnEventType
  - pv\_common\_types.h, 48
- TPVCmnExclusivePtr
  - pv\_common\_types.h, 48
- TPVCmnInterfacePtr
  - pv\_common\_types.h, 48
- TPVCmnMIMETYPE
  - pv\_common\_types.h, 48
- TPVCmnResponseType
  - pv\_common\_types.h, 48
- TPVCmnSDKInfo, 46
  - TPVCmnSDKInfo, 46
- TPVCmnSDKInfo
  - iDate, 46
  - iLabel, 46
  - operator=, 46
  - TPVCmnSDKInfo, 46

TPVCmnSDKModuleInfo  
    pv\_common\_types.h, [48](#)  
TPVCmnUUID  
    pv\_common\_types.h, [48](#)