- `PVMFErrProcessing`

PVPlayer engine specific error codes are listed below.

PVPlayer Engine Error Code	Error Description
PVPlayerEngineErrSourceInvalid	User provides an invalid data source for multimedia playback
PVPlayerEngineErrSourceInit	Error when initializing data source
PVPlayerEngineErrSource	General non-fatal error from the data source
PVPlayerEngineErrSourceFatal	General fatal error from the data source
PVPlayerEngineErrSourceNoMediaTrack	Data source contains no media track for playback
PVPlayerEngineErrSinkInvalid	User provides an invalid data sink for multimedia playback
PVPlayerEngineErrSinkInit	Error when initializing data sink
PVPlayerEngineErrSink	General non-fatal error from the data sink
PVPlayerEngineErrSinkFatal	General fatal error from the data sink
PVPlayerEngineErrNoSupportedTrack	No supported media track for playback was found
PVPlayerEngineErrDatapathInit	Error when initializing the datapath and its nodes
PVPlayerEngineErrDatapath	General non-fatal error from the datapath or its nodes
PVPlayerEngineErrDatapathFatal	General fatal error from the datapath or its nodes
PVPlayerEngineErrSourceMediaDataUnavailable	Data source ran out of media data
PVPlayerEngineErrSourceMediaData	General error in the data source's media data
PVPlayerEngineErrSinkMediaData	General error in the data sink's media data
PVPlayerEngineErrDatapathMediaData	General error in the datapath's or its nodes' media data
PVPlayerEngineErrSourceShutdown	Error when shutting down the data source
PVPlayerEngineErrSinkShutdown	Error when shutting down the data sink
PVPlayerEngineErrDatapathShutdown	Error when shutting down the datapath and its nodes

13.3 Error Code Translation and Error Chain

When components below PVPlayer engine (i.e. PVMF nodes) report an error, PVPlayer engine receives and processes the error, and if the error needs to be reported to the user of PVPlayer SDK, the error as one of PVMF status code is passed up. A PVMF status code is passed up so the user can expect a limited set of error codes.

But for users of PVPlayer SDK that can handle more specific error information from PVPlayer engine and components below PVPlayer engine, the error from PVPlayer engine contains a linked list that shows the trail of error message from the originator of the error to the error message that was received by PVPlayer engine and the error message generated by PVPlayer engine. To understand all this error message

information would require the user to have access to the context specific error codes used by the generator of error message at each level. To allow the context to be determined, the error list entry is based on PVMFErrorInfoMessageExtension, which provides access to the UUID for the error context with the error code. PVMFErrorInfoMessageExtension also allows the miscellaneous error message information (non-error code) to be retrieved as well via other extension interfaces. The user would need to understand the extension interface and know the UUID for the extension interface. PVMFErrorInfoMessageExtension derives from PVInterface so it contains a reference counter so memory used could be allocated by the originator of the error message but the memory would be deallocated properly when the all the users have finished using the message. For more information on PVMFErrorInfoMessageExtension including its features and usage, refer to its design document.

To illustrate how this error code translation is performed in PVPlayer SDK, an example with PVPlayer engine and data source nodes are shown below.

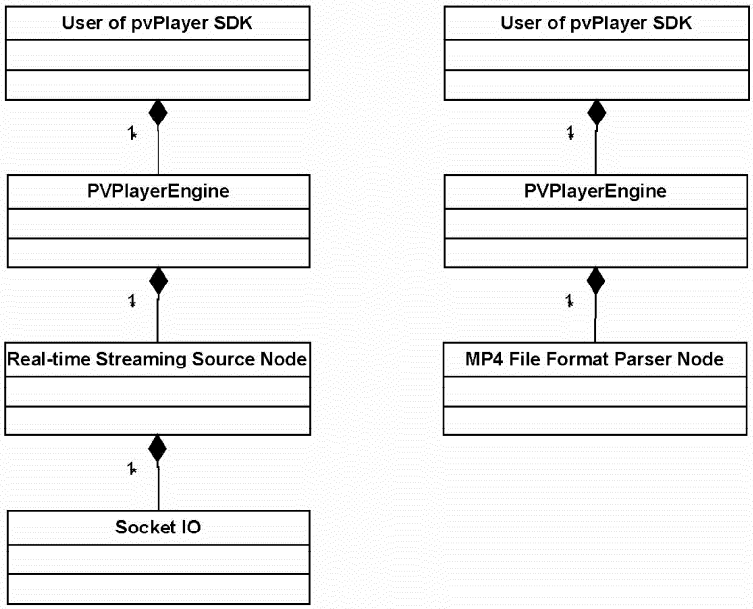

Figure 16: Class Diagram of Error Chain

The structure on the left shows the error propagation path for real-time streaming source when the error originates in network socket interface. The structure on the right shows the error propagation path for local file source when the error originates in the MP4 file format parser.

For the streaming case, the error originates in the socket IO level and real-time streaming source node receives a basic error message with an error code from socket IO. The streaming source node prepends its own error code to the socket IO error using PVMFErrorInfoMessageExtension's error chaining feature. Streaming source node packages its streaming specific error code with associated error details and sends an error event to PVPlayer engine. When PVPlayer engine receives the error event from the streaming source node, PVPlayer engine's own error code is prepended to the streaming source node's error and sends an error event to the PVPlayer SDK user. So the user of PVPlayer SDK receives PVMF error code and error messages from PVPlayer engine, streaming source node, and socket IO node which resulted in the engine level error. The streaming source node error message also provides more

information about the error than just an error code by including RTSP error code and error strings if available. The diagram below shows how the error event, PVAsyncEvent, received by PVPlayer SDK would contain the error messages from PVPlayer engine, streaming source node, and socket IO node.

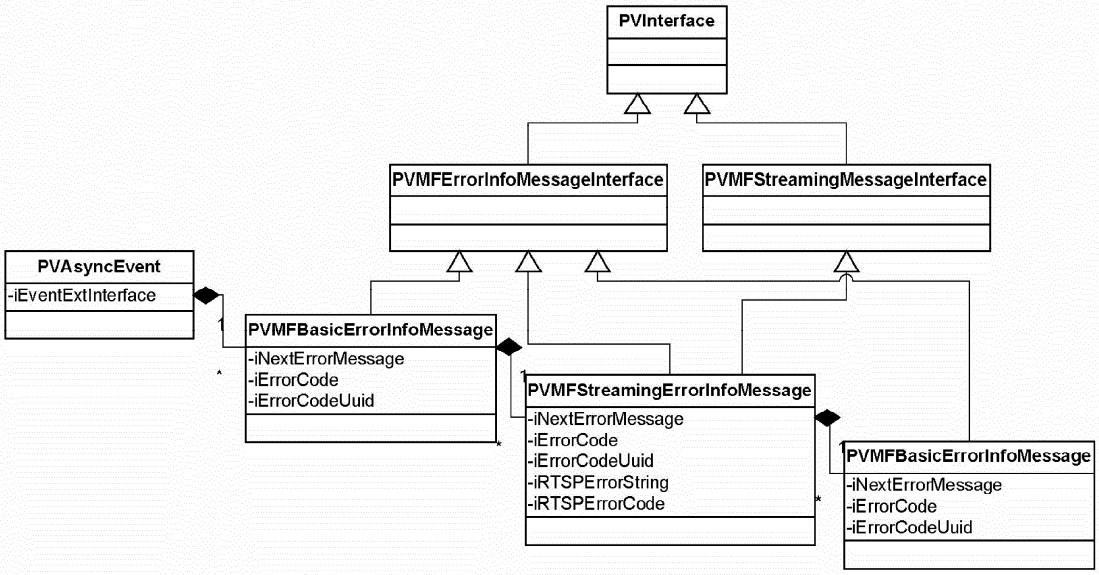

Figure 17: Streaming Error Event and Chain

For the local file case, the error originates in the MP4 file format parser node. In addition to the error code, the error message generated by the parser includes the MP4 atom where the error occurred. This error message is passed up to PVPlayer engine. PVPlayer engine then reports the error event to the PVPlayer SDK user with its own error code. The diagram below shows the error event and chain that the PVPlayer SDK user would receive for this case.

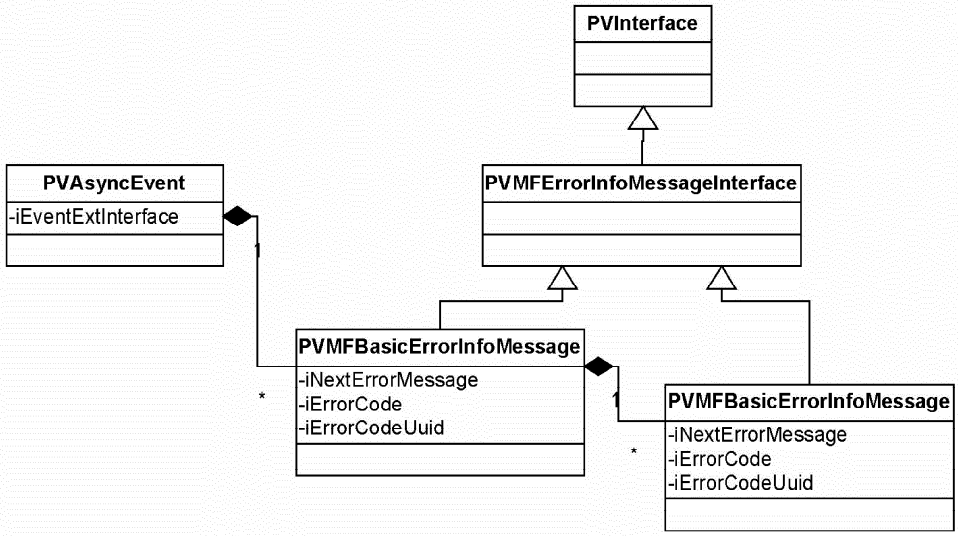

Figure 18: MP4 File Parsing Error Event and Chain

13.4 Typical Errors in Command Response

If a PVPlayer SDK API command fails, the failure is reported with error information in the CommandCompleted() callback. The following tables (one per API) list typical errors reported in response to API commands, the cause of the error, and expected handling by the user of PVPlayer SDK. As stated before, if an error is reported, the PVPlayer SDK user should check the PVPlayer engine state before performing any error handling of its own. Some errors might not be major or fatal and do not require any error handling.

AddDataSource()

Error code	Likely Cause	Expected Handling
PVMFErrInvalidState	Not in idle state. Wrong state to call AddDataSource()	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErrArgument	Passed in player data source is invalid	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErrNotSupported	Specified player data source (format type) is not supported in this PVPlayer SDK.	Command is rejected but engine does not go into error state. No error handling needed
PVMFErrNotSupported, PVPlayerErrSourceInit	Source node does not support the required extension interface	Check engine state to see if async error handling is occurring or not. Cannot use that particular data source..
PVMFErrNoMemory	Required amount of memory not available in engine	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFErrNoMemory, PVPlayerErrSourceInit	Required amount of memory not available in source node.	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFFailure	General failure code. Components are not behaving as expected.	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.

Init()

Error code	Likely Cause	Expected Handling
PVMFErrInvalidState	Not in idle state. Wrong state to call Init()	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErrNoMemory, PVPlayerErrSourceInit	Required amount of memory not available in source node.	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.

PVMFErrResource, PVPlayerErrSourceInit	Error while initializing the source (e.g. file parsing error, file corrupt). Check error message for more specific info if available	Check engine state to see if async error handling is occurring or not. Remove the data source if needed.
PVMFFailure	General failure code. Components are not behaving as expected.	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFErrLicenseRequired, PVPlayerErrSourceInit	Authorization license needed to initialize the specified source	Should acquire a license (via player's license acquisition interface or other means) before calling Init() again.
PVMFErrAccessDenied, PVPlayerErrSourceInit	Rights management does not allow playback of the specified source.	Check engine state to see if async error handling is occurring or not. Remove the data source if needed.

AddDataSink()

Error code	Likely Cause	Expected Handling
PVMFErrInvalidState	Not in initialized state. Wrong state to call AddDataSink()	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErrArgument	Passed in player data sink is invalid	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErrNotSupported	Specified player data sink (format type) is not supported in this PVPlayer SDK.	Command is rejected but engine does not go into error state. No error handling needed
PVMFErrNoMemory	Required amount of memory not available in engine	Command is rejected but engine does not go into error state. No error handling needed but should shutdown the engine.

Prepare()

Error code	Likely Cause	Expected Handling
PVMFErrInvalidState	Not in initialized state. Wrong state to call Prepare()	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErrNotReady	No player data sink added yet.	Command is rejected but engine does not go into error state. Add at least one valid player data sink before calling Prepare() again.
PVMFErrNotSupported	Previously specified player data sink is not supported	Check engine state to see if async error handling is occurring or not. Remove the unsupported data sink before calling Prepare() again.
PVMFErrNoMemory	Required amount of memory not available in engine	Check engine state to see if async error handling is occurring or not. Should not continue and should

		shutdown the engine.
PVMFErrResourceConfiguration	Datapath could not created with specified source and sinks	Command is rejected but engine does not go into error state. Change the source and/or sinks.
PVMFErr..., PVPlayerErrSinkInit	Extension interface for file output sink node could be obtained	Check engine state to see if async error handling is occurring or not. Pass in a sink node instead of using the file output sink node.
PVMFErr..., PVPlayerErrSourceFatal	Source node reported a fatal error in response to one of the following node commands: Prepare, QueryDataSourcePosition, SetDataSourcePosition, or Start	Check engine state to see if async error handling is occurring or not. Should not playback this source.
PVMFErr..., PVPlayerErrDatapathInit	One datapath encountered error whiling initializing the datapath (e.g. setting up and connecting decoder and sink nodes)	Check engine state to see if async error handling is occurring or not. Should not continue. Check specific error codes for cause of error.
PVMFErr..., PVPlayerErrDatapathFatal	One datapath encountered error whiling starting data flow (i.e. calling node Start() on decoder and/or sink node).	Check engine state to see if async error handling is occurring or not. Should not continue. Check specific error codes for cause of error.
PVMFFailure	General failure code. Components are not behaving as expected.	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.

Start()

Error code	Likely Cause	Expected Handling
PVMFErrInvalidState	Not in prepared state. Wrong state to call Start()	Command is rejected but engine does not go into error state. No error handling needed.

SetPlaybackRange()

Error code	Likely Cause	Expected Handling
PVMFErrInvalidState	Wrong state to call SetPlaybackRange()	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErrArgument	Passed in reposition parameter is invalid (e.g. position value)	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErrNotSupported	Specified reposition parameter is not supported.	Command is rejected but engine does not go into error state. No error handling needed

PVMFErr..., PVPlayerErrSourceFatal	Source node reported a fatal error in response to one of the following node commands: QueryDataSourcePosition or SetDataSourcePosition	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFErr..., PVPlayerErrSink	Sink node reported an error in response to SkipMediaData() command.	Repositioning during playback did not succeed properly but playback is still occurring. Stop playback first before continuing.
PVMFFailure	General failure code. Components are not behaving as expected.	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.

Pause()

Error code	Likely Cause	Expected Handling
PVMFErrInvalidState	Not in started state or already auto-paused due to source underflow. Wrong state to call Pause()	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErr..., PVPlayerErrDatapathFatal	One datapath encountered error when pausing the datapath (e.g. node pause command failed on decoder and/or sink node).	Check engine state to see if async error handling is occurring or not. Check specific error codes for cause of error. Check resulting state after error handling completes before continuing
PVMFErr..., PVPlayerErrSourceFatal	Source node reported a fatal error in response to the node pause command	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFFailure	General failure code. Components are not behaving as expected.	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.

Resume()

Error code	Likely Cause	Expected Handling
PVMFErrInvalidState	Not in paused state or is in paused state due to auto-pause. Wrong state to call Resume()	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErr..., PVPlayerErrDatapathFatal	One datapath encountered error when resuming the datapath (e.g. node start command failed on decoder and/or sink node).	Check engine state to see if async error handling is occurring or not. Check specific error codes for cause of error. Check resulting state after error handling completes before continuing
PVMFErr..., PVPlayerErrSourceFatal	Source node reported a fatal error in response to one of the following node command:	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling

	QueryDataSourcePosition, SetDataSourcePosition, or Start	completes before continuing
PVMFErr..., PVPlayerErrSink	Sink node reported an error in response to SkipMediaData() command.	Repositioning when resuming did not succeed properly but playback has resumed. Stop playback first before continuing.
PVMFFailure	General failure code. Components are not behaving as expected.	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.

Stop()

Error code	Likely Cause	Expected Handling
PVMFErrInvalidState	Not in started, paused, or prepared state. Wrong state to call Stop()	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErr..., PVPlayerErrDatapathFatal	One datapath encountered error when stopping the datapath (e.g. node stop command failed on decoder and/or sink node).	Check engine state to see if async error handling is occurring or not. Check specific error codes for cause of error. Check resulting state after error handling completes before continuing
PVMFErr..., PVPlayerErrDatapathShutdown	One datapath encountered error when tearing down and resetting the datapath.	Check engine state to see if async error handling is occurring or not. Check specific error codes for cause of error. Check resulting state after error handling completes before continuing
PVMFErr..., PVPlayerErrSourceFatal	Source node reported a fatal error in response to the node stop command	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFFailure	General failure code. Components are not behaving as expected.	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.

RemoveDataSink()

Error code	Likely Cause	Expected Handling
PVMFErrInvalidState	Not in initialized state. Wrong state to call RemoveDataSink()	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErrArgument	Passed in data sink is invalid	Command is rejected but engine does not go into error state. No error handling needed.
PVMFFailure	Specified data sink does not match an existing datapath.	Command is rejected but engine does not go into error state. No error handling needed.

Reset()

Error code	Likely Cause	Expected Handling
PVMFErrInvalidState	Not in initialized state. Wrong state to call Reset()	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErr..., PVPlayerErrSourceShutdown	Source node reported a fatal error to node reset command.	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFFailure	General failure code. Components are not behaving as expected.	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.

RemoveDataSource()

Error code	Likely Cause	Expected Handling
PVMFErrInvalidState	Not in idle state. Wrong state to call RemoveDataSource()	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErrArgument	Passed in player data source is invalid	Command is rejected but engine does not go into error state. No error handling needed.

GetMetadataKeys()

Error code	Likely Cause	Expected Handling
PVMFErrInvalidState	Wrong state to call GetMetadataKeys()	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErrArgument	One or more passed-in parameter is invalid or there are no nodes with metadata interface	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErrNoMemory	Required amount of memory not available in engine	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFFailure	General failure code. Components are not behaving as expected.	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.

GetMetadataValues()

Error code	Likely Cause	Expected Handling
PVMFErrInvalidState	Wrong state to call GetMetadataValues()	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErrArgument	One or more passed-in parameter is invalid or there are no nodes with metadata interface	Command is rejected but engine does not go into error state. No error handling needed.
PVMFErrNoMemory	Required amount of memory not	Check engine state to see if async error

	available in engine	handling is occurring or not. Should not continue and should shutdown the engine.
PVMFFailure	General failure code. Components are not behaving as expected.	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.

13.5 Typical Error Events

PVPlayer SDK errors that are not encountered when processing a PVPlayer SDK API command (e.g. error during playback) will be reported via the PVErrorEventObserver as an unsolicited event. The following table lists typical error events reported, the cause of the error, and expected handling by the user of PVPlayer SDK. As stated before, if an error is reported, the PVPlayer SDK user should check the PVPlayer engine state before performing any error handling of its own. Some errors might not be major or fatal and do not require any error handling.

Error code	Likely Cause	Expected Handling
PVMFErrCorrupt, PVPlayerErrSourceMediaData	Invalid media data detected in the source node (e.g. invalid sample in file)	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFErrOverflow, PVPlayerErrSourceMediaData	Memory buffer overflowed in the source node (e.g. buffer to hold media data is not big enough)	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFErrResource, PVPlayerErrSourceMediaData	Error detected in an underlying resource used by the source node (e.g. file parser error)	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFErrProcessing, PVPlayerErrSourceMediaData	Error occurred in source node while processing media data (e.g. could not send on output port)	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFErrUnderflow, PVPlayerErrSourceMediaUnavailable	Media data ran out unexpectedly in the source node (e.g. reached end of file)	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFErrNoResources, PVPlayerErrSourceFatal	Underlying resource needed by source node not available (e.g. socket connection not available for streaming)	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFErrResourceConfiguration, PVPlayerErrSourceFatal	Underlying resource used by source node has a configuration error	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.

PVMFErrTimeout, PVPlayerErrSourceFatal	Timeout occurred in the source node or underlying resource (e.g. socket connection timeout in streaming)	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFErrNoMemory, PVPlayerErrSourceFatal	Required amount of memory not available in the source node	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFErrCorrupt, PVPlayerErrDatapathMediaData	Invalid media data detected in one of the datapath node (e.g. invalid bitstream in the decoder node)	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFErrOverflow, PVPlayerErrDatapathMediaData	Memory buffer overflowed in one of the datapath node (e.g. buffer to hold media data is not big enough)	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFErrUnderflow, PVPlayerErrDatapathMediaData	Media data ran out unexpectedly in the one of the datapath node	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFErrProcessing, PVPlayerErrDatapathMediaData	Error occurred in datapath node while processing media data (e.g. could not send on output port)	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFErrTImeout, PVPlayerErrDatapathFatal	Timeout occurred in the datapath node or underlying resource (e.g. hardware audio decoder timed out)	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFErrNoResources, PVPlayerErrDatapathFatal	Underlying resource needed by datapath node not available (e.g. DSP video memory not available)	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFErrResourceConfiguration, PVPlayerErrDatapathFatal	Underlying resource used by datapath node has a configuration error (e.g. video decoder cannot handle the specified dimension)	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFErrResource, PVPlayerErrDatapathFatal	Error detected in an underlying resource used by a datapath node (e.g. audio decoder encountered unrecoverable error)	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFErrNoMemory, PVPlayerErrDatapathFatal	Required amount of memory not available in the one of the datapath or one of the nodes	Check engine state to see if async error handling is occurring or not. Should not continue and should

	in the datapath.	shutdown the engine.
PVMFErrCorrupt, PVPlayerErrSinkMediaData	Invalid media data detected in sink node (e.g. invalid decoded data for video display)	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFErrOverflow, PVPlayerErrSinkMediaData	Memory buffer overflowed in the sink node (e.g. buffer to hold media data is not big enough)	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFErrUnderflow, PVPlayerErrSinkMediaData	Media data ran out unexpectedly in the sink node (e.g. audio device underflows and then errors out)	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFErrProcessing, PVPlayerErrSinkMediaData	Error occurred in sink node while processing media data (e.g. could not receive on input port)	Check engine state to see if async error handling is occurring or not. Check resulting state after error handling completes before continuing
PVMFErrTimeout, PVPlayerErrSinkFatal	Timeout occurred in the sink node or underlying resource (e.g. audio device timed out)	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFErrNoResources, PVPlayerErrSinkFatal	Underlying resource needed by sink node not available (e.g. audio device not available)	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFErrResourceConfiguration, PVPlayerErrSinkFatal	Underlying resource used by sink node has a configuration error (e.g. audio device cannot handle the specified sampling rate)	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFErrResource, PVPlayerErrSinkFatal	Error detected in an underlying resource used by the sink node (e.g. video render device encountered unrecoverable error)	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFErrNoMemory, PVPlayerErrSinkFatal	Required amount of memory not available in the sink node	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.
PVMFFailure	General failure code. Components are not behaving as expected.	Check engine state to see if async error handling is occurring or not. Should not continue and should shutdown the engine.

13.6 Fault Detection, Handling and Recovery

Fault is an incorrect and unexpected system state. PVPlayer SDK will try to detect faults based on the information available. If the fault is avoidable or recoverable, PVPlayer engine will report the fault as an informational event and try to continue operation. If the fault is unrecoverable, PVPlayer engine will go into an error state and report the error to the layer above. The layer above will then need to reset or destroy the PVPlayer engine instance to resume operations from the fault. In some faults like memory allocation failure, PVPlayer engine will allow the leave to propagate up to the layer above to be trapped unless the PVPlayer SDK requirement for the platform does not allow leaving.

14 Usage Scenarios

To illustrate how PVPlayer SDK would be typically used, this section will present several PVPlayer SDK usage scenarios. Scenarios will cover different sources, playback features, and error conditions. The PVPlayer SDK represented in the scenarios will support all the features, but the interface would be the base level OSCL-based interface and all underlying nodes are OSCL-based software nodes. The lifelines in the sequence diagram will be limited to PVPlayer SDK interface and the user of the SDK unless the scenario calls for other object lifelines.

14.1 Instantiating PVPlayer SDK

The sequence diagram shows how the PVPlayer engine object is created via the factory component. After instantiation, PVPlayer engine is in IDLE state.

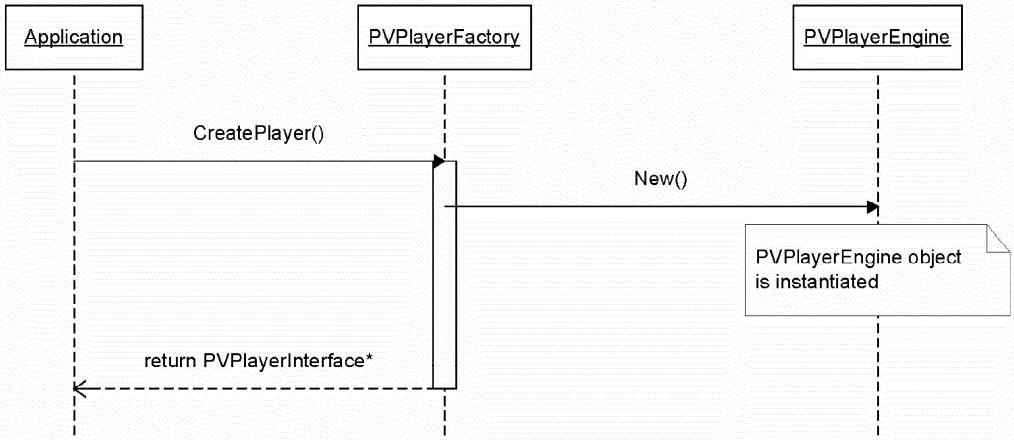

Figure 19: Sequence Diagram for Creating PVPlayer

14.2 Shutting down PVPlayer SDK

The sequence diagram shows how the PVPlayer engine object is destroyed via the factory component. PVPlayer engine object should be in IDLE state to properly destroy it.

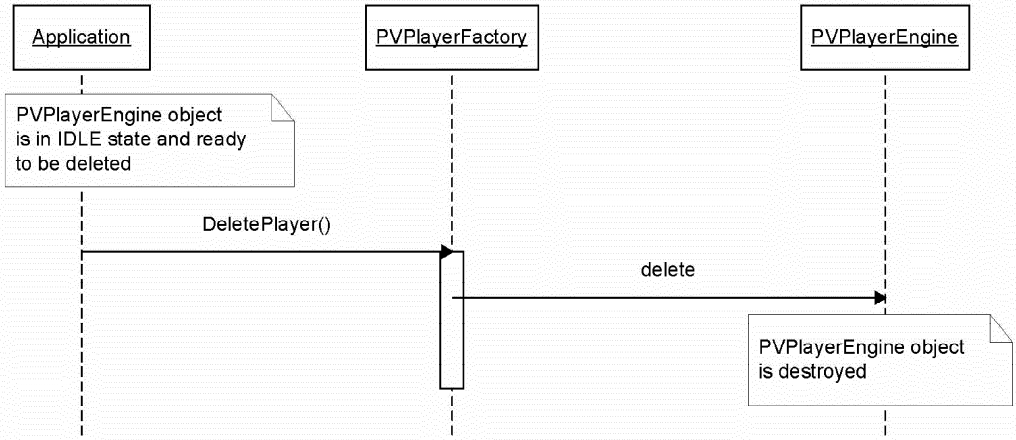

Figure 20: Sequence Diagram for Deleting PVPlayer

14.3 Open a Local MP4 File, Play and Stop

In this scenario, a local MP4 file containing audio and video tracks is specified as the data source, audio and video data sinks are added, playback is started, and then stopped after some time.

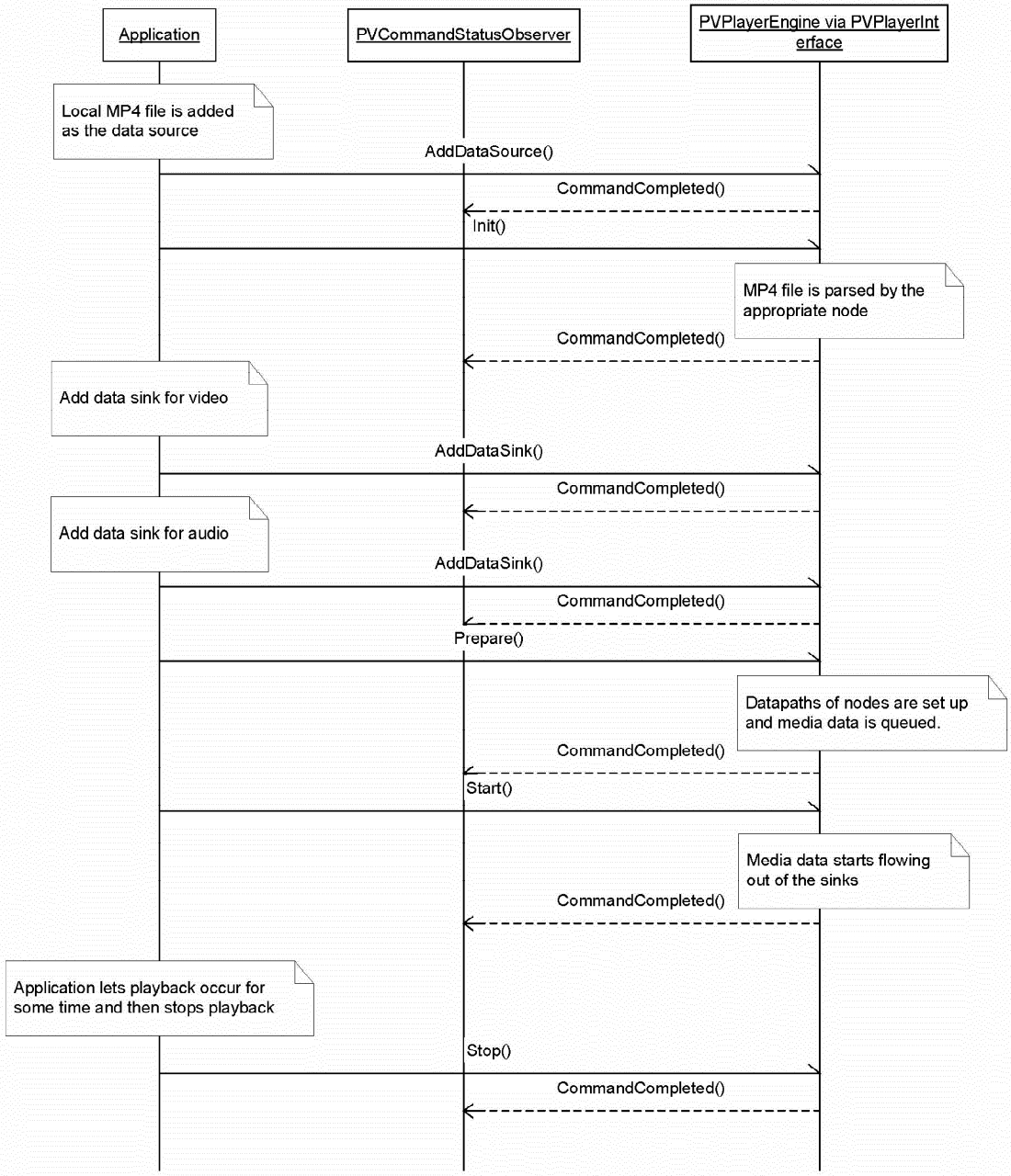

Figure 21: Open a Local MP4 File, Play and Stop

14.4 Open a RTSP URL, Play and Stop

In this scenario, a streaming source containing audio and video media tracks is specified by a RTSP URL. Then an audio and a video data sinks are added, playback started and then stopped after some time.

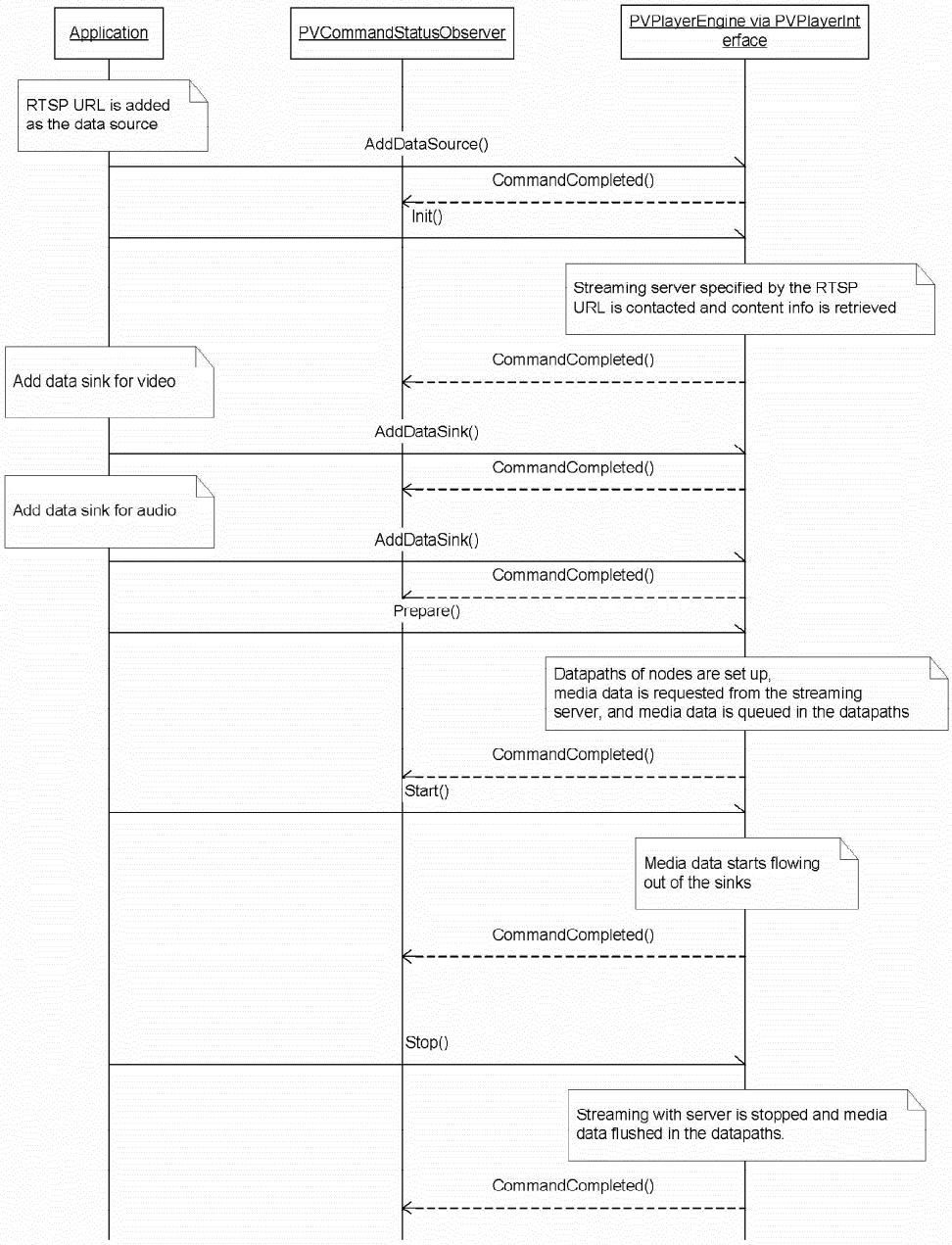

Figure 22: Open a RTSP URL, Play and Stop

14.5 Play a Local File Until End of Clip

This scenario is similar to a prior local file scenario but instead of ending playback due to the user calling a control API, the playback pauses since the end of clip is reached. The user still needs to call Stop() after the playback is automatically paused to stop the playback.

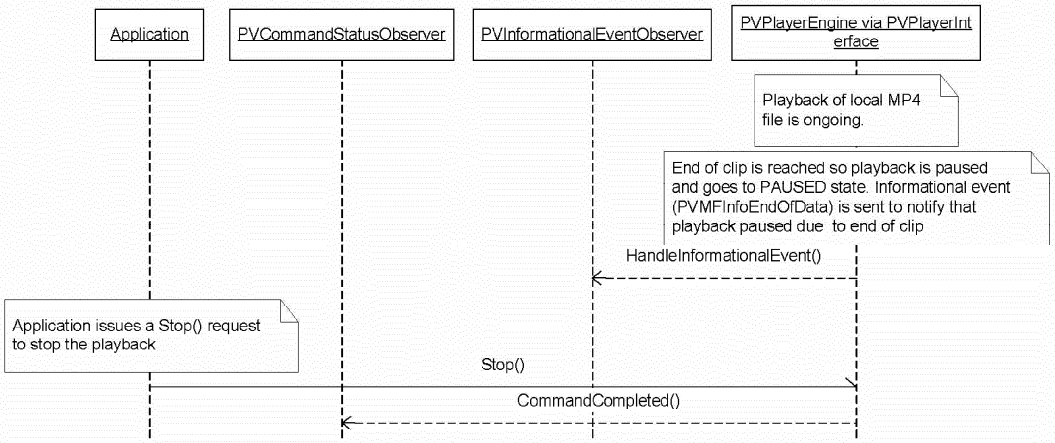

Figure 23: Play a Local File Until End of Clip

14.6 Play a Local File, Stop and Play Again

This scenario shows how to re-play a clip after it has been played and then stopped. PVPlayer engine goes to INITIALIZED state after Stop() command completes so Prepare() and Start() are called again to restart playback.

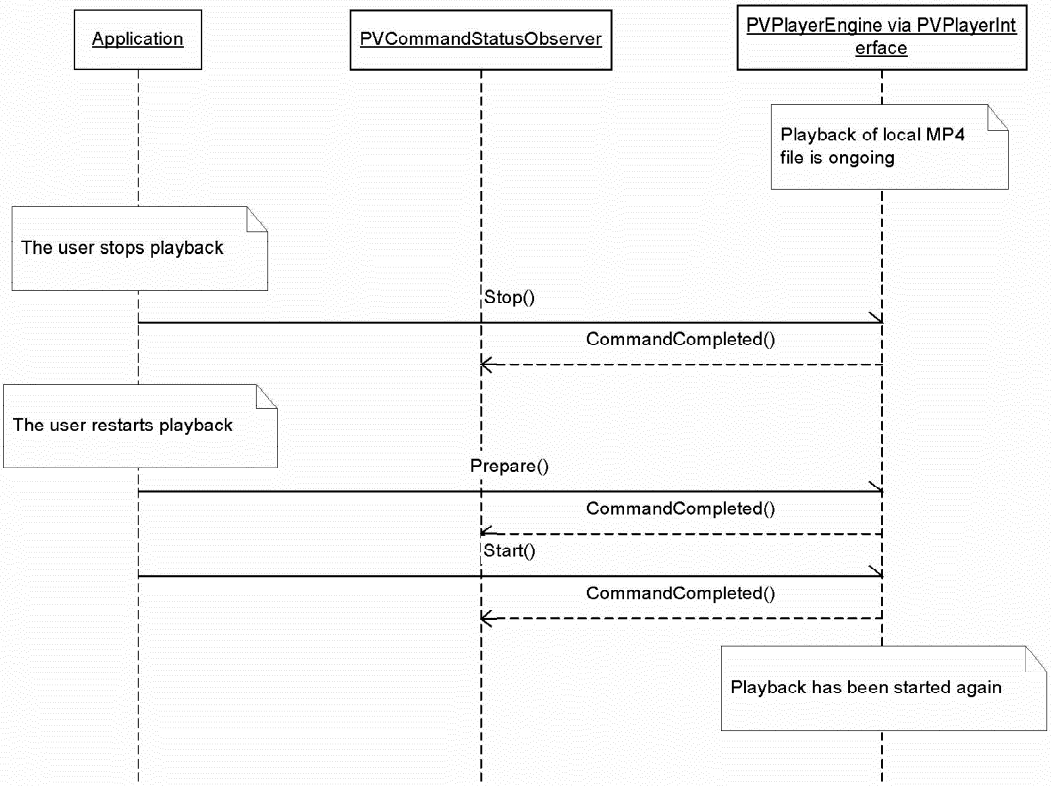

Figure 24: Play a Local File, Stop and Play Again

14.7 Play a local file, stop, open another file, and play

This scenario shows how to open another clip for playback after playing the current clip. The proper sequence for closing the current clip is shown.

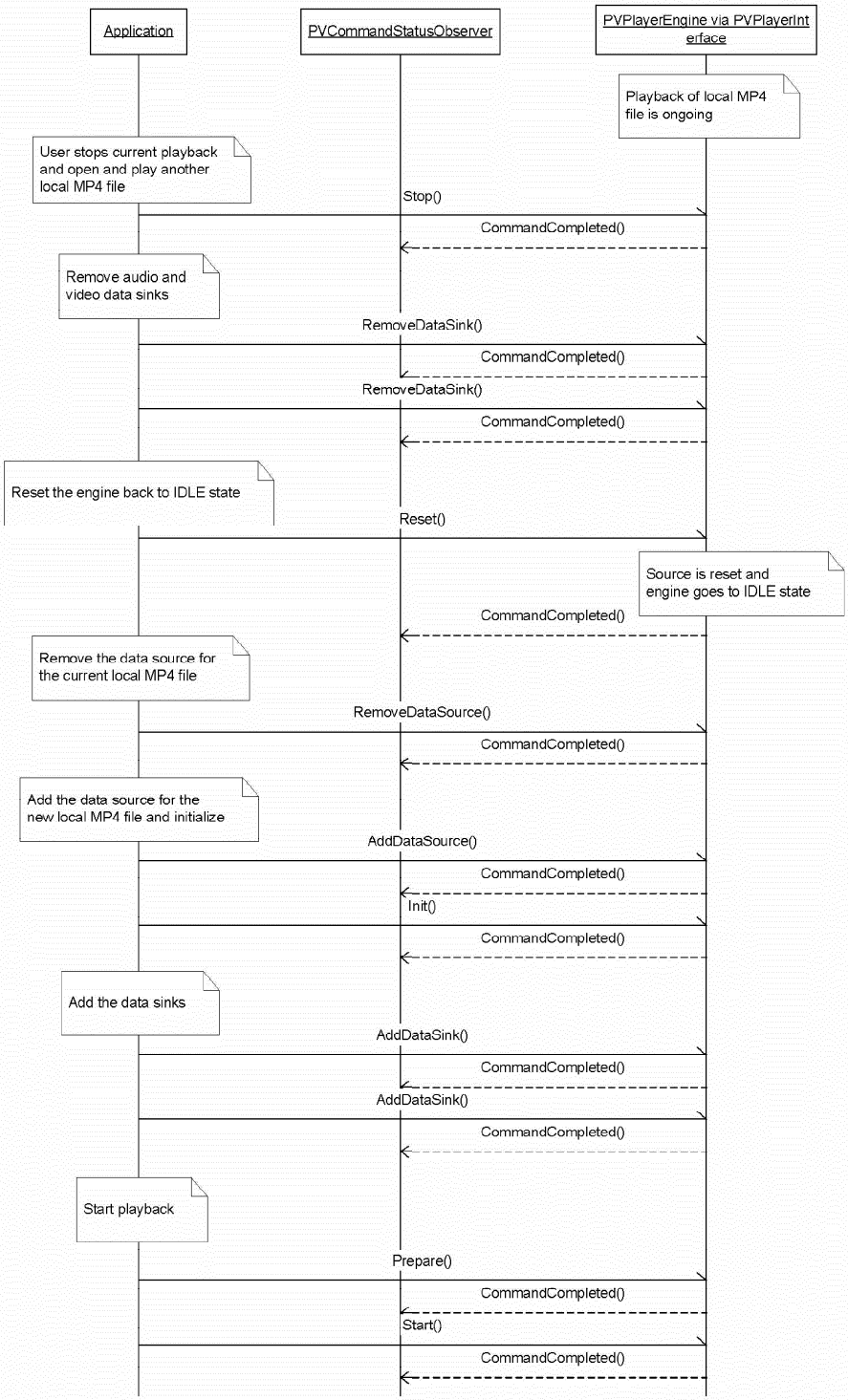

Figure 25: Play a local file, stop, open another file, and play

14.8 Play a local file, pause, and resume

In this scenario, a playback is paused and then playback is resumed.

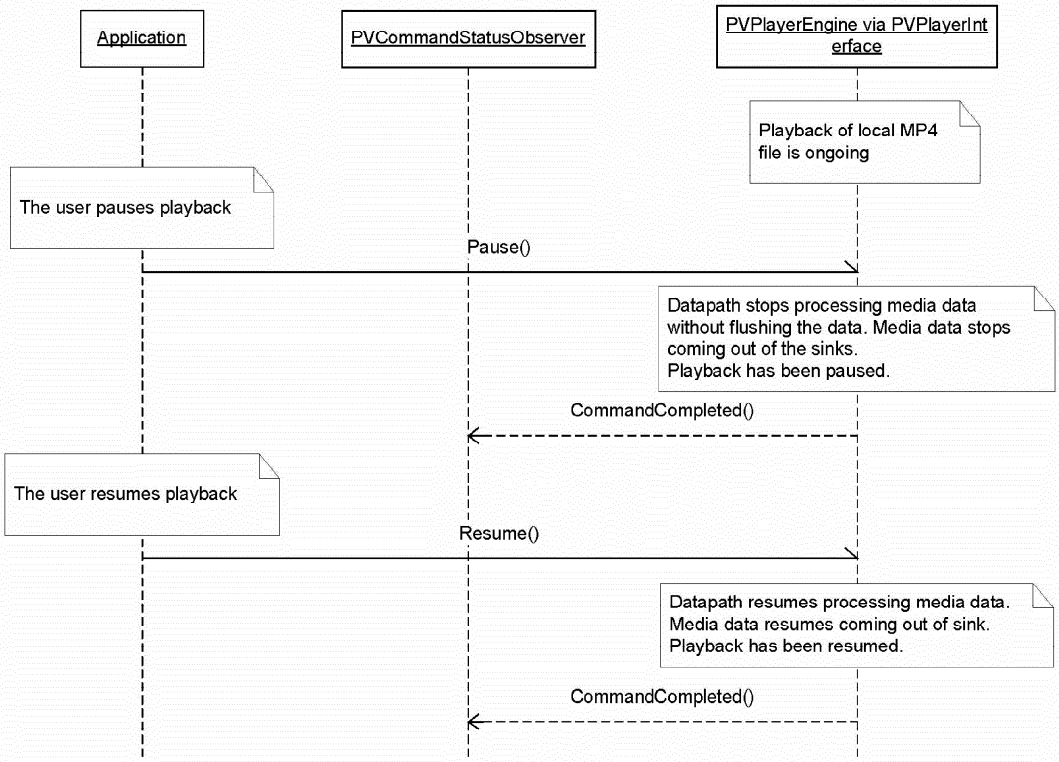

Figure 26: Play a local file, pause, and resume

14.9 Play a local file, pause, and stop

In this scenario, a playback is paused and then stopped when paused.

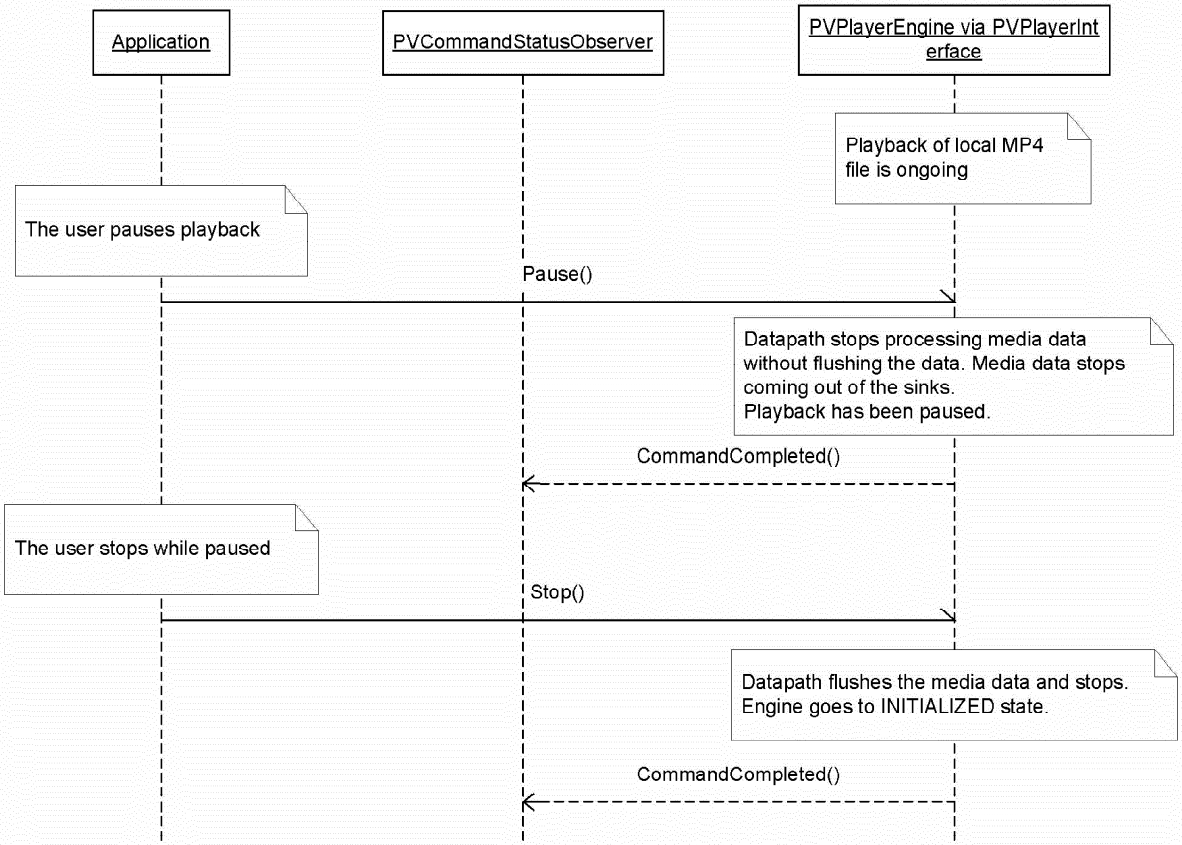

Figure 27: Play a local file, pause, and stop

14.10 Playback of DRM Protected Contents

When playing back DRM-protected content, the interaction between the application and player engine is very similar to playback of non-protected content with the exception of the handling of license acquisition and some of the events associated with license acquisition. If the license for a piece of content is available and valid, then playback will happen automatically using the same sequence of steps as with non-protected content. If no valid license is available, then the behavior depends on whether this piece of content had a previous valid license that expired. In most cases the engine simply notifies the application that acquisition of a valid license is required and relies on the application to decide, possibly through user interaction, if that should be done.

However, in some cases, it may be desirable to have the player engine automatically attempt to acquire the new valid license if the existing one has expired. An example would be some form of subscription service that the user has joined. In that case, the process is streamlined by having the player engine automatically attempt to acquire the license when necessary. A property stored along with the license determines whether it should be automatically acquired, so this behavior is only relevant after a license for a piece of content is acquired for the first time. The following subsections describe the interaction between the application and player engine for the different scenarios of handling DRM content.

14.10.1 Preparation to Play DRM Protected Contents

Before any DRM content can be played, the appropriate CPM (Content Policy Manager) plug-in modules must be registered for usage by player engine. For information on CPM plug-ins, refer to the PV CPM Plug-in Programming Guide.

CPM plug-ins are registered with a factory function and a MIME string, using the PVMFCPMPluginFactoryRegistryClient class. For each plug-in, the application instantiates the factory and passes the factory plus the MIME string to the registry. During playback of protected content, the player engine will check the registry and instantiate the plug-ins using their factory functions as needed. In applications that support multiple types of DRM, there will be multiple CPM plug-in modules. In a multi-DRM scenario, the application can register all available plug-ins without concern for which plug-in will be used for a particular piece of content, since player engine makes the determination during playback.

The CPM plug-ins can be registered anytime before playback of protected content occurs. The plug-ins will remain registered until the registry client session is closed. When the client session is closed, all plug-ins registered during that session are automatically cleaned up and removed. An application could register all plug-ins before creating player engine then cleanup plug-ins after player engine is destroyed.

Besides registering plug-ins, the application needs to set a Boolean flag in the local source data to tell player engine that the content may be DRM protected. In some cases, the application may not know whether a particular piece of content is protected or not. If there is any possibility that it is protected, the Boolean flag should be set to true.

The sequence diagram below shows the use of the CPM plug-in factory registry to register CPM plug-ins for use by player engine.