Content Policy Manager Developer's Guide
OHA 2.0, rev. 2
Oct 6, 2009

References

1. Open Mobile Alliance. http://www.openmobilealliance.org/
2. PacketVideo Corp. *PVPlayer SDK Developer's Guide*. OHA 1.0 Rev 1. Oct 15, 2008.
3. PacketVideo Corp. *PV Data Stream Interface.* Version 1.1. Sept 21, 2005.

Table of Contents

List of Figures

1 Introduction

The OpenCore Content Policy Manager (CPM) is a subsystem that provides a generic framework for controlling and enforcing content usage rules. A popular use-case of the content policy manager involves providing access to files protected with some form of Digital Rights Management (DRM). There are many different DRM schemes in use today and this will certainly continue to evolve over time, so the focus of the CPM design is to provide a flexible framework for accommodating new plugins. Some of the more popular examples of specific DRM schemes include those defined by the Open Mobile Alliance (OMA) standards body and those will be referenced throughout this document to provide concrete examples. The CPM itself does not directly implement DRM algorithms and protocols, but instead provides a framework for developing and registering plugins that provide those implementations. Through interfaces for authentication, authorization, and access, the CPM provides methods for the content consumer to gain access to the media data from the content provider.

While managing and providing access to DRM-protected content is a major use-case, the scope of the content policy management extends beyond that domain. The CPM allows for rules that extend beyond a single piece of content. For example, another common use-case for the CPM involves the implementation of parental controls to restrict access to content based on ratings and user profile. Since the content management rules can be layered, it is possible to implement a wide range of rules by combining a set of plugins with limited scope. In summary, the purpose of the CPM is to provide well-defined interfaces between consumers and providers/managers of content for mutually establishing the identities of the parties involved, authorizing content usage, and providing access to the content.

The next section describes some of the details of popular DRM schemes to provide motivation for the design of the CPM. After this background information, the details of CPM are described starting with the typical steps involved in a session between a content consumer and the CPM.

1 Background on DRM Schemes

This section presents some background on two broad classes of DRM schemes to motivate the design of the CPM and its interfaces and semantics. The differences relate to the specifics of how the encryption is applied to content. References to OMA DRM will be used to provide concrete examples.

1.1 Wrapped DRM

DRM schemes in this class protect content by wrapping it with a uniform encryption layer across all of the data. This method of encryption allows the DRM to be handled in a fairly transparent way from the point of view of the file parsers, but it does have drawbacks for streaming scenarios since the encryption is not necessarily aligned with meaningful boundaries in the underlying content (e.g., video frames). As a specific example, OMA 1.0 provides following levels of protection to any piece of content:

- **Combined Delivery**
  The associated rights object is delivered along with the multimedia content. Forward lock is a special case of combined delivery where in the rights object is absent and default rights apply (which in this case amounts to a "forward-lock" on the content). In case of combined delivery the content and the associated rights object are delivered over secure means, and hence the content itself need not be encrypted, as per OMA 1.0 specification. However it is not uncommon for this content to be encrypted regardless of the security of the transport.

- **Separate Delivery**
  The associated rights object is delivered separately from the multimedia content. In this case the content is typically encrypted and is called super distributable since it can be delivered over any non-secure means without compromising DRM rules. The final consumer of the content needs to procure rights for the same over some secure means prior to content consumption.

Encryption used in OMA1.0 is applied uniformly to the entire content using a symmetric cipher operating on blocks of data. The encryption wraps the entire file and is completely unaware of the content format (i.e., it is content agnostic). The result is that any interpretation of the content is not possible without decryption. OMA 1.0 does not address DRM requirements of streaming applications.

1.1 Embedded DRM

DRM schemes in this class protect content by embedding encrypted data within a plaintext container file format or protocol. This case is not transparent to the file format or protocol parsers because certain parts of the data need to be decrypted. If a full parser were embedded within the DRM agent it may be possible to make the DRM transparent to a player, but it would require parsing the content multiple times as well as potentially maintaining multiple parsers. An integration of this type could be handled as an instance of the wrapped DRM mentioned in the previous section. However, for cases where the file and protocol parsing will be handled in one place, a different interaction between the media player and DRM agent is needed. The format used for continuous media (e.g., audio and video) in OMA 2.0 DRM is an example of this type of embedded DRM.

OMA 2.0 provides protection mechanisms for:
- Local Playback
- Progressive Download
- Streaming

OMA 2.0 defines a new file format for protected content that is based on ISO BaseMedia file format. This format defines containers that contain DRM-specific information in addition to containing encrypted media itself. These containers themselves are not encrypted. As a result some sections of the file can be interpreted by the content consuming application without relying on the DRM agent. However, the format parser must also be involved in the decryption process because it is necessary to understand the format in order to extract the data for decryption.

1.1 DRM Content Consumption

At the highest level, the content consumer and DRM agent interaction can be broken down into Control interactions and Data interactions. The following two sections discuss these interactions in more detail.

1.1.1 Control Interactions

Control interactions involve actions such as obtaining the rights object and checking for content rights. In case of the wrapped DRM schemes, the content consumer does not need to provide any additional information to the DRM agent to perform rights management or procurement as shown in Figure 1. In the case of the embedded DRM schemes, the content consumer needs to provide information from the DRM container within the multimedia file to the DRM agent in order for it to perform rights management or procurement as shown in Figure 2.

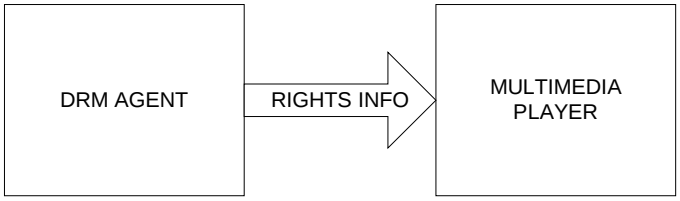

Figure 1: Control Interaction for Wrapped DRM

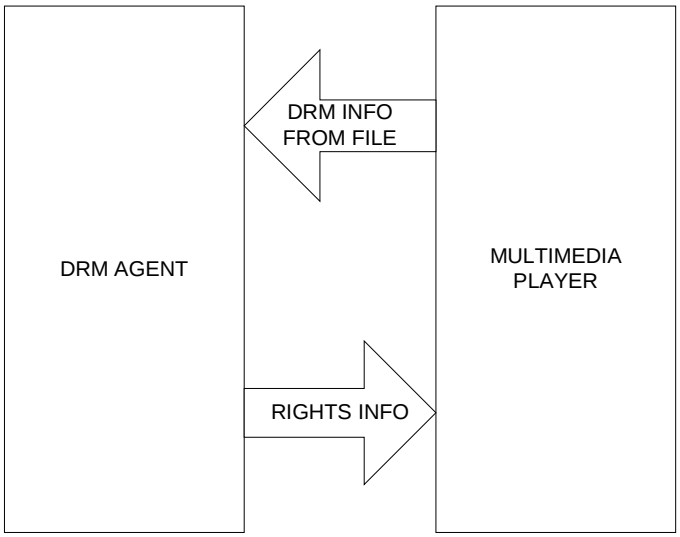

Figure 2: Control Interaction for Embedded DRM

1.1.2 Data Interactions

Data interactions for DRM content consumption for playback involve content read and seek operations. In the case of the wrapped DRM schemes, since the content is encrypted uniformly over the top of any data format, the access to the plaintext data relies entirely on the DRM agent and can be implemented in a transparent way. DRM agents for the wrapped DRM schemes can often be integrated below the low-level file I/O interfaces for this reason. The data flow is illustrated in Figure 3.

For embedded DRM schemes, the files are only partially encrypted and require that the consumer is aware of the format and must interact with the DRM agent directly to obtain the plain-text content. In that case the consumer is between the file I/O and the DRM agent as shown in Figure 4.

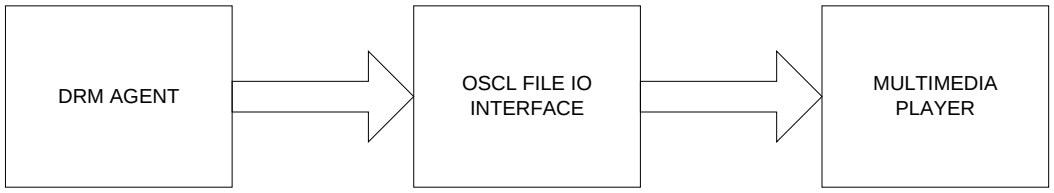

Figure 3: Data Interactions for Wrapped DRM.

Figure 4: Data Interactions for Embedded DRM

2 Content Policy Manager

The role of the Content Policy Manager (CPM) is to aggregate the services related to content management and usage rules from the active plugins and make them available to the content consumer. For example DRM content access is a service, parental control is another service etc. Every one of these services could require a plugin. Figure 5 illustrates how the interaction is structured with the CPM facilitating the communication between the consumer, the media player in this case, and the service plugins.

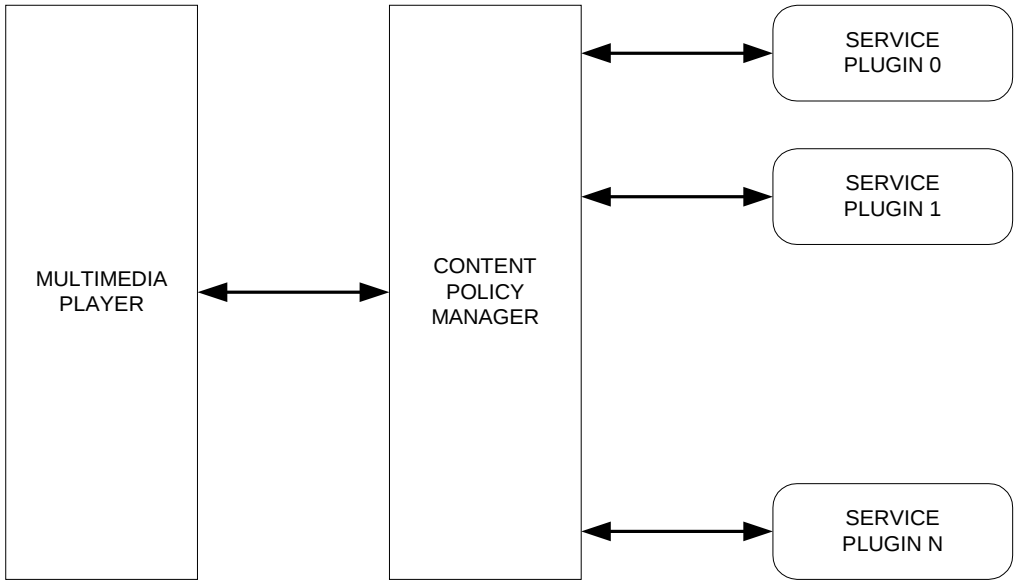

Figure 5: Interaction between Player, CPM and CPM Plugins

Content access could involve interactions with one or more plugins. For example, parental control plugin would be involved for all content access. For DRM content access, both the DRM plugin and parental control plugin would be involved. Each plugin may provide the following interfaces:

- Authentication interface
- Authorization interface
- Content access interface factory

An authentication interface is always provided. Authorization and content access interfaces are provided based on a content handle. If the content handle corresponds to media that the plugin does not really care about, then it would not provide authorization and access interfaces.

2.1 Content Access

This section describes the use of the content policy manager module to perform content access. Any access, procurement and consumption of content can be broken down into following stages:

1) Acquire resources
2) Open a session
3) Register content handle
4) Get content type

5) Specify intent
6) Perform DRM content access
7) Specified Intent Complete
8) Close the DRM Session
9) Release DRM resources

Each one of these stages is described below:

2.1.1 Acquire Resources

Any initialization or configuration of the various registered plug-ins (viz. DRM Agent) are done in this stage.

Rationale: In many DRM agent implementations it must be instantiated and initialized once before any interactions are possible. Then all subsequent interactions should happen with that instance.

2.1.2 Open a Session

The user of the CPM must establish a unique session with the CPM for content access. Establishment of a session involves authenticating the user with the various registered plug-ins, through authentication interfaces of all registered plug-ins. A session is said to have been established when all the registered plug-ins accept the identity of the user.

Rationale: In some usage scenarios, the users of the DRM agent would need to be authenticated before resources provided by the DRM agent can be accessed. This authentication could be based on a digital signature provided by the user to the DRM agent, such that the DRM agent can verify that the user is an authorized application in the platform to use DRM resources.

2.1.3 Register Content Handle

Content consumer would provides a reference to the content that needs to be accessed, such as file name or file handle, etc, and CPM would in turn identify a list of active plug-ins that are related to this content. Any plugin that provides an authorization interface is considered active for this content. Only one among the many active plug-ins would provide content access to perform these actions. If there were more than one active plugin that is capable of providing access, then CPM would select the most suitable one.

2.1.4 Get Content Type

CPM informs the content consumer what type of DRM scheme is used to protect the requested content. In general this could evolve into CPM providing content-recognition capability for various types of content.

2.1.5 Specify Content Usage

The content consumer must specify how the content will be used to the CPM in order to check the intended usage against the usage rules. For example, a playback engine might report the intention to PLAY the content, while a download engine might report the intention to DOWNLOAD the content. The CPM passes the specified content usage values to each of the active plugins for authorization. All of the active plugins must authorize the specified content usage in order for access to be provided.

Rationale: In most usage scenarios, the users of the DRM content would need some sort of authorization to be performed before their intent is accepted. Hence the need execute intent asynchronously.

2.1.6 Perform Content Access

Once the user has established a session with CPM, then the user can perform the required media manipulations, namely read, seek, write etc.

2.1.7 Content Usage Complete

Once the media consumption is completed, the consumer signals the CPM that the current usage of the content, specified earlier,is complete. Any future use of the content requires stating intent again.

2.1.8 Close Session

When the multimedia application is done with the CPM, the session is closed.

2.1.9 Release Resources

Remove or unload registered plugin instances.

2.2 Use-Cases for Protected Content Consumption

This section describes the interaction between various PVPlayer SDK modules and CPM during consumption of protected contents.

2.2.1 Local Playback Of Wrapped DRM Content

This section describes the interaction between the user of the content manager (the fileparser node) and the content manager, and the latter's interaction with the underlying DRM agent on the device, during local playback of content protected with a wrapped DRM scheme.

The wrapped DRM content are typically block encrypted and in order to interpret any part of it, the data must be decrypted first. In the use case described above, the content manager is used to handle the rights procurement and expect the file I/O calls to handle the decryption of the content behind the scenes. Both the content manager and the file I/O modules need to interact with the DRM APIs. This approach is inevitable since most DRM agents using a wrapped DRM scheme integrate below the file system interface.

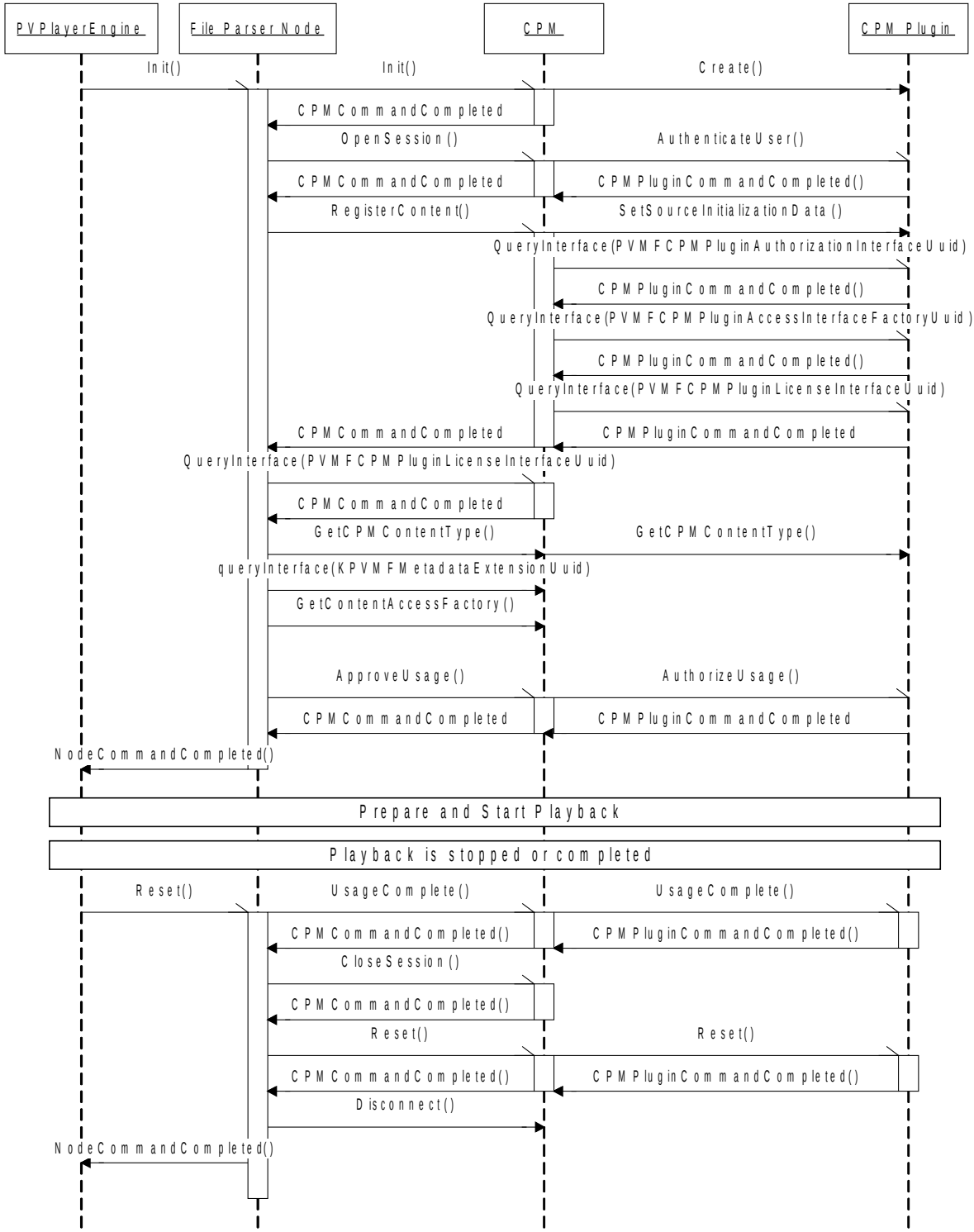

Figure 6: Local Playback of Wrapped DRM Content

2.2.2 Local Playback of Embedded DRM Content

In case of an embedded DRM scheme, the file I/O operations are decoupled from the decryption of the content. The metadata is stored in the clear, and the player would need to open the content file to parse for the license procurement information, then pass the information to the DRM agent for content access.

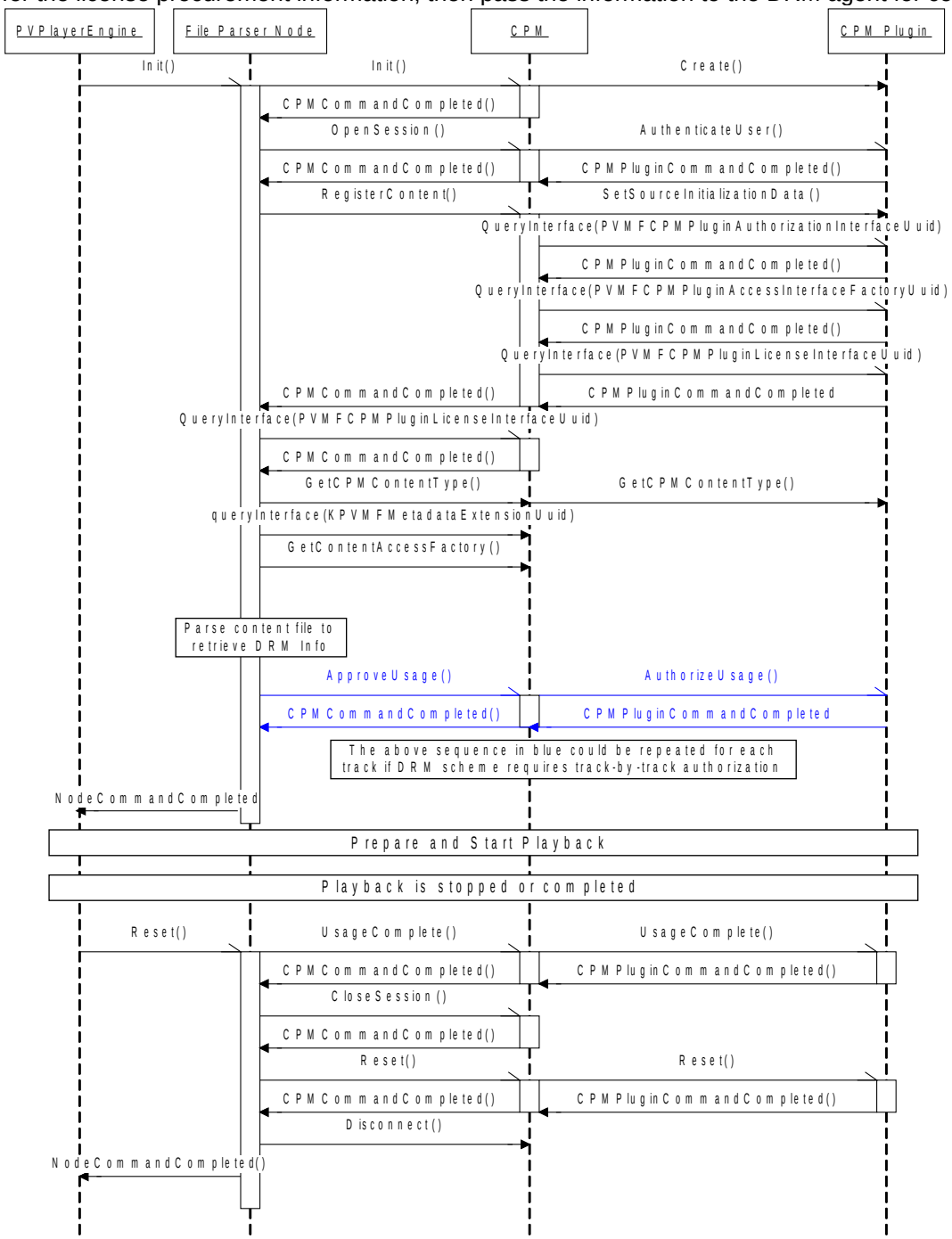

Figure 7: Local Playback of Embedded DRM Content

3 Content Policy Manager Plugin

A content policy manager plugin is an entity that enables the content policy manager to enforce certain rules with regard to content access. The kind of rules fall into three broad categories:

- User Authentication (the term “user” is used in a general sense to include any entity that desires multimedia functionality from the player and does not necessarily refer to the consumer)
- Authorization for a particular piece of content based on expressed intent (intent being things like play, pause, seek back, seek forward, copy, save etc)
- Access to content (provide means for the content consumer to do operations like open, read, seek etc on the content)

From a conceptual standpoint a plugin can be visualized as an entity that provides means to implement these rules. From a implementation standpoint this would translate to plugins being classes/objects that implement certain well defined interfaces.

3.1 CPM Plugin Registration

In order for the CPM plug-ins to be usable by the OpenCore framework, they must be registered by the application. There is a registration class called PVMFCPMPluginFactoryRegistryClient to do this, and it is defined in the file pvmf_cpmplugin_factory_registry.h. To use the registry, the application first opens a client session then registers one or more CPM plug-ins. Each plugin is registered by providing a unique MIME string identifying the plugin, plus a pointer to an implementation of a factory function. The registered plug-ins will remain usable by the PV software until the application software closes its plugin registry session.

Each MIME string identifying a CPM plugin must begin with the string “X-CPM-PLUGIN” followed by a “/” followed by a unique plugin identifier. For example, a plugin implemented by Acme company for a DRM scheme called X47 might use an identifier “X-CPM-PLUGIN/ACME-DRM-X47”.

The plugin factory function must implement the PVMFCPMPluginFactory interface. If the plugin requires constructor parameters, those parameters can be contained within the factory implementation. The PV software will call the factory function’s Create and Destroy methods as needed to instantiate the CPM plug-ins in order to access protected content.

3.2 Authentication Interface

The authentication interface provides methods for mutual identification and verification of the user and the content provider. Authentication plays an important role in the implementation of policies and usage rules. For example, the content provider may need the user identity in order to evaluate an authorization request and in many cases the identity needs to be established in a secure way so that it can be trusted.

The establishment of identity is also important for situations where some degree of trust is required. The module obtaining access to the raw media from DRM-protected content needs to be trusted to follow the specified usage rules. Conversely that module may want to establish the identity of the content provider before supplying any sensitive information. Therefore, the authentication interface provides for optional challenge requests so each side can verify the identity of the other with a higher degree of confidence.

3.2.1 Expressing User Profile

It is possible to register a user profile with a parental control plugin in such a way that the content policy manager could restrict content access with the help of this plugin. Typically a user profile would contain the following:

- Rating Entity and corresponding allowed Rating Criteria (for example this would be: Rating Entity: BBFC(which is a four character code for *British Board of Film Classification*) and a rating criteria of “12” (No one under 12 years of age may see a "12" film or rent or buy a "12" video).
- URL for age/content/user verification

In order to do this, a generic extensible way of expressing user profile needs to be devised. One of the ways of expressing user profile could be through key-value pairs.

```
“x-pvmf/cpm/userprofile/rating-source; valtype=char*”
      rating-source could be BBFC, MPAA, OFLC etc
“x-pvmf/cpm/userprofile/authorized-rating; valtype=fourCC”
      authorized-rating is dependent on the rating entity.
      Example: PG13, NC-17 etc
“x-pvmf/cpm/userprofile/authorizationURL; valtype=char*”
```

3.1 Authorization Interface

3.1.1 Expressing Content Usage

In order to establish a DRM session with the CPM, the player module needs to specify its intent towards the multimedia file. Typically, intent could be a combination of any of the following:

- Preview
- Play
- Pause
- Rewind
- Seek Forward
- Seek Backward
- Print
- Execute
- Download
- Save

In the future, other use-scenarios may be required, so it is important that the method of expressing content usage is easily extensible. One extensible method, which is used in other parts of the PacketVideo architecture where extensibility is required, is based on key-value pairs. The key portion of the key-value pair is a character string holding an extended MIME string, which allows a great deal of flexibility and practically limitless extensibility. The value portion of the key-value pair can be one of several data types depending on the valtype parameter in the key string. Since the content usage may involve a number of the options shown in the list above, it is most efficient to represent the usage as a packed bit array for compactness. The following figure shows an example of how it would be defined.

```
“x-pvmf/cpm/use-profile; valtype=bitarray32; name=pvmf-cpm-usage;
version=1.0”
where intent value could be a combination of any of the following bit masks:
      BITMASK_PVMF_CPM_PREVIEW = 0x00000001,
      BITMASK_PVMF_CPM_PLAY = 0x00000002,
      BITMASK_PVMF_CPM_PAUSE = 0x00000004,
      BITMASK_PVMF_CPM_REWIND = 0x00000008,
      BITMASK_PVMF_CPM_SEEK_FORWARD = 0x00000010,
      BITMASK_PVMF_CPM_SEEK_BACK = 0x00000020,
      BITMASK_PVMF_CPM_PRINT = 0x00000040,
      BITMASK_PVMF_CPM_DOWNLOAD = 0x0000080,
      BITMASK_PVMF_CPM_SAVE = 0x00000100,
      BITMASK_PVMF_CPM_EXECUTE = 0x00000200
```

Any future content usage would be specified using an updated version of the current bitmask or a new bitmask.

3.1 Content Access Interface Factory

The CPM plugin provides content access indirectly through the content access interface factory method. The idea is that the factory can be used to create potentially multiple instances of the content access interfaces and is used to get instances of a data stream interface (i.e., PVMIDataStreamSyncInterface) for basic read support and a decryption interface (i.e., PVMFCPMPluginAccessUnitDecryptionInterface) for decrypting blocks of data for the embedded DRM content. Figure 8 illustrates the interactions with the access interfaces.

3.2 CPM Plugin Implementation

From an implementation standpoint, a plugin is an ActiveObject that implements some or all of the above described interfaces and runs within the context of PV Player SDK’s scheduler. The plugin itself may be created in a different thread and passed in through the plugin registry mechanism into the PV Player SDK thread. Hence connect and disconnect APIs have been defined to explicitly add and remove the plugin to and from a scheduler’s queue. Any implementation that desires to use the plugin instance across multiple threads must use two different instances of the same and the third party APIs that the plugin integrates with need to take care of thread safety issues, if any.

3.2.1 Reference Implementation

PV provides a reference CPM plugin “pvoma1passthruplugin” stands for pass-thru mode of DRM OMA 1.0 to facilitate third party plugin implementers. The plugin is located under “\pvmi\content_policy_manager\plugins\oma1\passthru”. The associated runtime dynamic loadable module wrapper “pvpvoma1passthrupluginterface” is located under “\modules\cpm_oma1_passthru”.

The reference plugin implements all CPM plugin interfaces such as AuthenticateUser() for Authenticate interface, AuthorizeUsage() for Authorization interface, and SetSourceInitializationData(), etc. for Content access interfaces. Since it is a simulated OMA 1.0 pass-thru mode plugin, it is designed to accept all content formats without any access restriction except ASF which needs Janus plugin.

PV provides source, makefiles, and a binary of the pre-built pvoma1passthurplugin and pvoma1passthrupluginterface modules:

File	Location	Description
libpvoma1passthruplugin.so	/system/lib	Pre-build binary version of pvoma1passthruplugin.so
pvmf_cpmplugin_passthru_oma1.h pvmf_cpmplugin_passthru_oma1_config.h pvmf_cpmplugin_passthru_oma1_factory.h pvmf_cpmplugin_passthru_oma1_types.h	/pvmi/content_policy_manager/plugins/oma1/passthru/include	Interface & definition of pre-build pvoma1passthruplugin.so
pvmf_cpmplugin_passthru_oma1.cpp	/pvmi/content_policy_manager/plugins/oma1/passthru/src	Source code of pre-build pvoma1passthruplugin.so
local.mk	/pvmi/content_policy_manager/plugins/oma1/passthru/build/make	Make file of the pre-build pvoma1passthruplugin.so
pvmf_oma1_passthru_plugin_module.cpp	/modules/cpm_oma1_passthru/src	Source code of pre-build pvoma1passthrupluginterface
local.mk	/modules/cpm_oma1_passthru/build/make	Make file of the pre-build pvoma1passthrupluginterface
pvdrm_oma1passthru.cfg	/modules/cpm_oma1_passthru/build/make to be copied to /system/etc	Config file contains dlls such as pre-build pvoma1passthruplugin
Pvoma1passthrupluginterface.so	/system/lib	Pre-build binary version of Pvoma1passthrupluginterface.so

To use pre-build pvoma1passthruplugin.so and pvoma1passthrupluginterface.so in test. Please copy the pvdrm_oma1passthru.cfg to the designated 'cfg' file folder such as /system/etc where runtime loadable module configuration files are to parsed by pvplayer engine.

- Modify the source code as wished for adding OMA 1.0 access rules.
- Re-build the library using the provided makefile
- Replace the pre-built binary with the new version

The library name should not be modified, since the PV code is pre-configured to look for the oma1 passthru modules using the existing name.

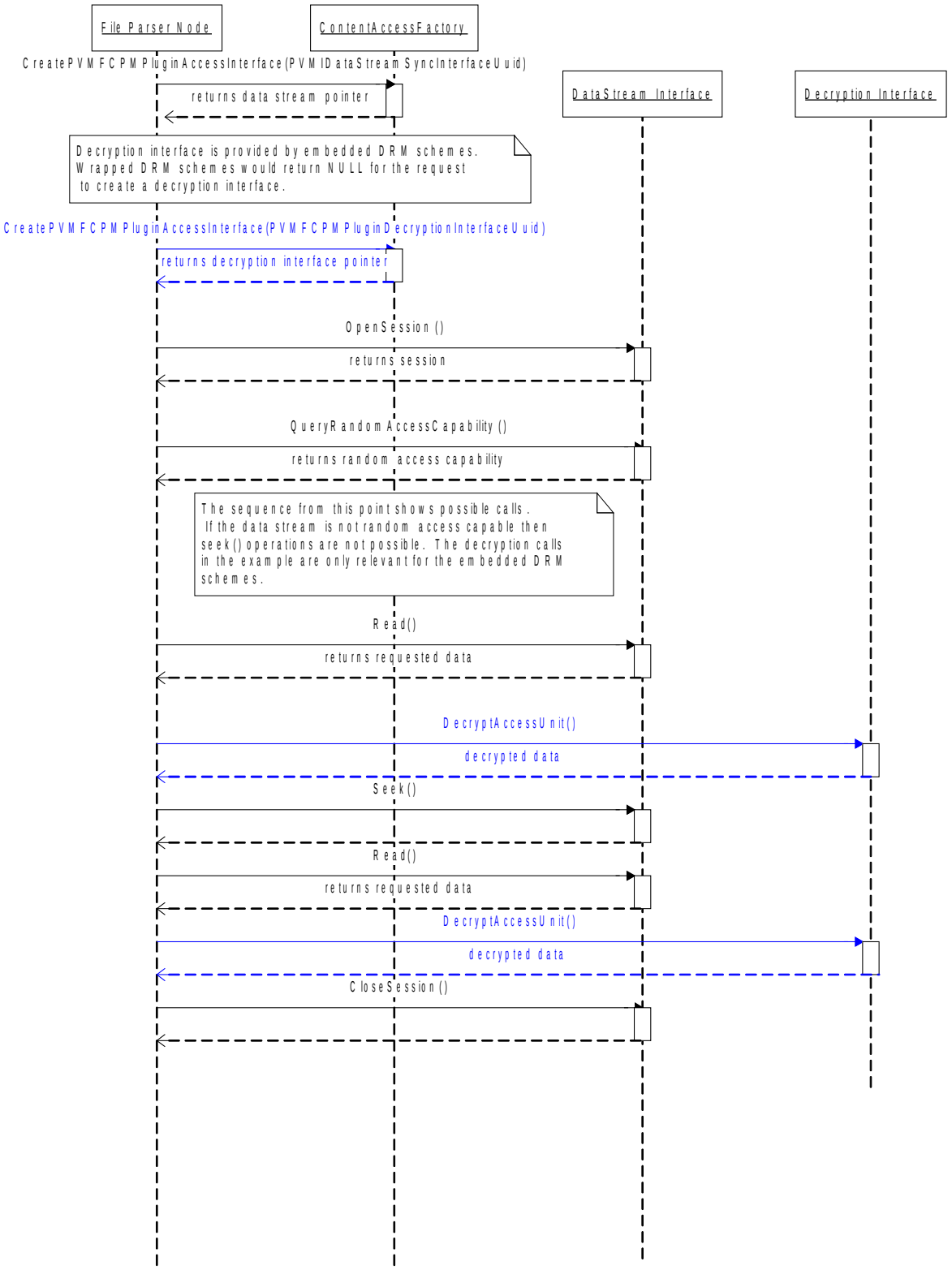

Figure 8: Interaction with the CPM Access Interfaces