DRM Content Format

Candidate Version 2.2 – 19 Apr 2011

Open Mobile Alliance

OMA-TS-DRM-DCF-V2_2-20110419-C

Contents

Figures

Tables

1. Scope

Open Mobile Alliance (OMA) specifications are the result of continuous work to define industry-wide interoperable mechanisms for developing applications and services that are deployed over wireless communication networks.

The scope of OMA "Digital Rights Management" (DRM) is to enable the distribution and consumption of digital content in a controlled manner. The content is distributed and consumed on authenticated devices per the usage rights expressed by the content owners. OMA DRM work addresses the various technical aspects of this system by providing appropriate specifications for content formats, protocols, and rights expression languages.

The scope for this specification is to define the content format for DRM protected encrypted media objects and associated metadata. This specification addresses the specific format mechanisms defined in the Release 2.2 "*Digital Rights Management*" specification [DRM-v2.2].

2. References

2.1 Normative References

[AES] "Recommendation of Block Cipher Modes of Operation", NIST, NIST Special Publication 800-38A,
URL:http://www.nist.gov/

[DRMCF-v2.2] "DRM Content Format V2,2", Open Mobile Alliance™, OMA-TS-DRM-DCF-V2_2,
URL:http://www.openmobilealliance.org/

[DRMREL-v2.2] "DRM Rights Expression Language V2.2", Open Mobile Alliance™, OMA-TS-DRM-REL-V2_2,
URL:http://www.openmobilealliance.org/

[DRM-v2.2] "Digital Rights Management V2.2", Open Mobile Alliance™, OMA-TS-DRM-DRM-V2_2,
URL:http://www.openmobilealliance.org/

[IOPPROC] "OMA Interoperability Policy and Process", Version 1.11, Open Mobile Alliance(tm), OMA-IOP-Process-V1_11,
URL:http://www.openmobilealliance.org/

[ISO14496-12] "Information technology — Coding of audio-visual objects – Part 12: ISO Base Media File Format", International Organization for Standardization, ISO/IEC 14496-12, Second Edition, April 2005

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997,
URL:http://www.ietf.org/rfc/rfc2119.txt

[RFC2234] "Augmented BNF for Syntax Specifications: ABNF", D. Crocker, Ed., P. Overell, November 1997,
URL:http://www.ietf.org/rfc/rfc2234.txt

[RFC2326] "Real Time Streaming Protocol (RTSP)", Schulzrinne H., Rao A. and Lanphier R., April 1998

[RFC2327] "SDP: Session Description Protocol", IETF RFC 2327, Handley M. and Jacobson V., April 1998

[RFC2392] "Content-ID and Message-ID Uniform Resource Locators", E. Levinson, August 1998,
URL:http://www.ietf.org/rfc/rfc2392.txt

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee et al, August 1998,
URL:http://www.ietf.org/rfc/rfc2396.txt

[RFC2616] "Hypertext Transfer Protocol -- HTTP/1.1", R. Fielding, et al, June 1999,
URL:http://www.ietf.org/rfc/rfc2616.txt

[RFC2630] "Cryptographic Message Syntax", R. Housley, June 1999,
URL:http://www.ietf.org/rfc/rfc2630.txt

[RFC3550] "RTP: A Transport Protocol for Real-Time Applications", Schulzrinne H. et al., July 2003

[TS26.244] "Transparent end-to-end Packet-switched Streaming Service (PSS); File Format", The Third Generation Partnership Project, Release 6, TS 26.244, Version 6.4.0,
URL:http://www.3gpp.org/

[ISO/IEC 13818-1] ISO/IEC 13818-1, Information Technology – Generic Coding of moving pictures and associated audio information: Systems,
URL:http://www.iso.org

[XBS-v1.1] "OMA DRM v2.0 Extensions for Broadcast Support". Version 1.1. Open Mobile Alliance™. OMA-TS-DRM_XBS-V1_1.
URL: http://www.openmobilealliance.org/

[IEC62455] "Internet Protocol (IP) and Transport Stream (TS) based Service Access", International Electrotechnical Commission, IEC 62455 ed2.0, October 2010
URL:http://www.iec.ch/

2.2 Informative References

[C.S0045] "Multimedia Messaging Service (MMS) Media Format and Codecs for cdma2000 Spread Spectrum Systems" ", Version 1.0, 3GPP2, C.S0045-0v1.0 Multimedia Messaging Service (MMS) Media Format and Codecs for cdma2000 Spread Spectrum Sys, tems
URL:http://www.3gpp2.org/

[C.S0050] "File Formats for Multimedia Specifications", 3GPP2, Version 1.0, C.S0050-0v1.0 3GPP2 File Formats for Multimedia Specifications,
URL:http://www.3gpp2.org/Public_html/specs/C.S0050-0_v1.0_121503.pdf

[DRMCF-v1] "DRM Content Format", Open Mobile Alliance™, Version 1.0, OMA-DRM-DRMCF-v1_0,
URL:http://www.openmobilealliance.org/

[DRMREL-v1] "DRM Rights Expression Language", Open Mobile Alliance™, Version 1.0, OMA-DRM-DRMREL-v1_0,
URL:http://www.openmobilealliance.org/

[DRM-v1] "Digital Rights Management", Open Mobile Alliance™, Version 1.0, OMA-Download-DRM-v1_0,
URL:http://www.openmobilealliance.org/

[ISO7498-2] ISO/IEC 7498: "Information processing systems -- Open Systems Interconnection -–- Basic Reference Model - Part 2: Security Architecture", International Organization for Standardization, ISO/IEC 7498

[TS26.234] "Transparent end-to-end packet-switched streaming service (PSS); Protocols and Codecs", , The Third Generation Partnership Project, Release 6, TS 26.234, Version 6.4.0,
URL:http://www.3gpp.org/

[WSP] "Wireless Session Protocol", WAP Forum™, WAP-230-WSP,
URL:http://www.openmobilealliance.org/

[OMADICT] "Dictionary for OMA Specifications", Version x.y, Open Mobile Alliance™, OMA-ORG-Dictionary-Vx_y, URL:http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

Box	A data structure derived from the “Box” definition in [ISO14496-12].
Composite Object	A Media Object that contains one or more Media Objects by means of inclusion e.g. DRM messages, zip files.
Confidentiality	The property that information is not made available or disclosed to unauthorised individuals, entities or processes. (From [ISO 7498-2])
Content	One or more Media Objects.
Content Issuer	The entity making content available to the DRM Agent; the entity whose Content is being Protected.
Content Object	A single piece of Content contained in a DCF data structure. A Content Object may be DRM Content or unprotected Content.
Continuous Media	Content which is inherently time-based, i.e. might have an implicit or explicit duration and requires multiple iterations of an algorithm to produce a continuous media experience to a User, such as video or audio.
Device	A Device is a user equipment with a DRM Agent. The Device MAY include a smartcard module (e.g. a SIM) or not depending upon implementation.
Discrete Media	Content that can be rendered with a single pass of an algorithm to interpret the media content, media that itself does not contain an element of time, such as still images or web pages.
DRM Agent	The entity in the Device that manages Permissions for Media Objects on the Device.
DRM Content	Media Objects that are consumed according to a set of Permissions in a Rights Object.
Enforced Advertising	The mechanism on advertisement content to be enforced for rendering according to the rules contained in RO or DCF.
Integrity	The property that data has not been altered or destroyed in an unauthorised manner.
Media Object	A digital work e.g. a ringing tone, a screen saver, a Java game or a Composite Object.
Permission	Actual usages or activities allowed (by the Rights Issuer) over DRM Content.
Rights Issuer	An entity that issues Rights Objects to OMA DRM Conformant Devices.
Rights Object	A collection of Permissions, Constraints and other attributes which define under what circumstances access is granted to, and what usages are defined for, DRM Content. All OMA DRM Conformant Devices must adhere to the Rights Object associated with DRM content.
User	The human user of a Device. The User does not necessarily own the Device.
Broadcast Program	A logical portion of a Broadcast Service with a distinct start and end time. In the case the Broadcast Program is not free-to-air, it can be offered individually for purchase, such as “Pay-Per-View”, or as part of a parent service (e.g. subscription service). A Broadcast Program may for example represent a movie, news show or soccer game.
Broadcast Service	A digital broadcast service delivered in an MPEG-2 transport stream consisting of a concatenation of Broadcast Programs, as defined in an MPEG-2 Program Map Table (PMT).
Interaction Channel	A bi-directional channel used to engage in communication protocols (such as DRM v2 ROAP) with other entities. The Interactive Channel can for example be used to request a Rights Object from a Rights Issuer.

3.3 Abbreviations

3GPP	3rd Generation Partnership Project
4CC	Four Character Code
AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
CEK	Content Encryption Key
CTR	Counter Mode
DCF	DRM Content Format
DRM	Digital Rights Management
HTTP	Hypertext Transfer Protocol
ISO	International Standards Organisation
IV	Initialisation Vector
MIME	Multipurpose Internet Mail Extensions
OMA	Open Mobile Alliance
PDCF	Packetised DRM Content Format
PSS	Packet switched Streaming Service
RFC	Request For Comments
RO	Rights Object
ROAP	Rights Object Acquisition Protocol
RTP	Real time Transport Protocol
RTSP	Real Time Streaming Protocol
SDL	Syntactic Description Language
SDP	Session Description Protocol
URI	Uniform Resource Indicator
URL	Uniform Resource Locator
MPEG2DCF	MPEG-2 Transport Stream DRM Content Format
STB	A Set Top Box. A device capable of receiving digital broadcast services contained in an MPEG-2 transport stream that may be delivered over cable, satellite, terrestrial, IP or any other medium. To access the digital broadcast services, a Set Top Box may or may not use a Conditional Access System. A STB may or may not be OMA DRM compliant.
CAS	Conditional Access System

4. Introduction

OMA "Digital Rights Management" (DRM) enables the distribution and consumption of digital content in a controlled manner by enabling Content Issuers to distribute DRM Content and Rights Issuers to issue Rights Objects for the DRM Content.

This specification defines the DRM v2.1 Content Format [DCF v2.1], and is one part of a set of OMA DRM 2.1 specifications. For a detailed discussion of the overall system architecture, please refer to [DRMARCH-v2.1]. For a detailed discussion of the Rights Expression Language that is used to construct the Rights Objects, please refer to [DRMREL-v2.1]. For a detailed discussion of the format and semantics of the cryptographic protocol, messages, processing instructions and certificate profiles to enable an end-to-end system for DRM protected content distribution, see the DRM part [DRMDRM-v2.1] of OMA DRM v2.1.

4.1 Version 1.0

The OMA DRM V1.0 specification provides some fundamental building blocks for a DRM system without addressing the complete security necessary for a robust, end-to-end DRM system that takes into account the need for secure distribution, authentication of Devices, revocation and other aspects.

Part of OMA DRM v1.0 is the DRM Content Format specification [DRMCF-v1].

4.2 Version 2.0

The main differences between OMA DRM v1.0 and OMA DRM v2.0 are significantly improved security and functionality.

Within OMA DRM v2.0, Media Objects are encrypted and packaged into a specific format, the DRM Content Format (DCF v2.0). The DCF can be delivered separately from an associated Rights Object, which contains the encryption key used to encrypt the Media Object.

In addition to encrypting the Media Object, the DRM v2.0 Content Format supports metadata such as

- Original content type of the media object
- Unique identifier for this DRM protected Media Object to associate it with rights
- Information about the encryption details
- Information about the rights issuing service for this DRM protected media object
- Extensions and other media type dependent metadata

The file format is extensible, so additional features may be added in the future while maintaining compatibility with the older versions. Compatibility with the OMA DRM v1.0 Content Format [DRMCF-v1] is not maintained by this specification, therefore a different MIME type is used.

There are two profiles of the DRM Content Format. One is used for Discrete Media (such as still images) and one for Continuous Media (such as music or video). The profiles share some data structures. Both profiles are based on a widely accepted and deployed standard format, the ISO Base Media File format [ISO14496-12], but the Discrete Media profile is meant to be an all-purpose format, not aiming for full compatibility with ISO media files.

The Content Issuer can decide which profile to use for their content, but in general, the profile for Continuous Media should be used for Continuous Media content, in order to create a harmonious user experience. The Discrete Media profile should be used for other types of content. To a User, the difference is that a file conforming to the Discrete Media profile looks like a DRM protected file, whereas a file conforming to the Continuous Media profile looks and functions like a media file to the outside.

4.2.1 Version 2.0.1

The most important DCF changes introduced in DRM v2.0.1 compared with DRM v2.0 are summarized in section 4.2.1 of the DCF Specification [DRMDCF v2.0] in the DRM 2.0.1 ERP. These changes are considered to require special consideration in implementation. Most of the identified changes are bug fixes which if not implemented correctly may result in interoperability problems between conformant and non-conformant devices. Companies with existing DRM 2.0 implementations should take careful consideration of these changes.

4.3 Version 2.1

OMA DRM v2.1 has been developed as a result of market feedback. The main differences between OMA DRM v2.0 and OMA DRM v2.1 are the addition of several features on top of OMA DRM v2.0, including:

- Metering, primarily intended for information gathering. By means of metering, actual content usage information can be provided to Rights Issuers, thereby enabling royalty collection based on actual usage of content.
- Content differentiation, defining a mechanism to control how content can be consumed. For example, this mechanism can prevent that a music track is used as ringtone.
- RO installation confirmation.
- Additional metadata, such as artist, title and genre
- Support for user editable metadata, in addition to content issuer defined metadata.
- A binary format for ROAP triggers to improve communication efficiency.
- New domain property (noConsumeAfter) to simplify “temporary sharing” business models
- RO upload functionality to enable users to upload Rights from their old device to a Rights Issuer so that these Rights can be downloaded to their new device.
- Improved extensibility for future versions.

The DRM 2.1 features have minimum impact on the DRM 2.0 architecture and are defined in a DRM 2.0 backward compatible manner. The DCF v2.1 specification extends DCF 2.0 so as to support above features.

4.4 Version 2.2

The main differences between OMA DRM v2.2 and OMA DRM v2.1 are the addition of several features on top of OMA DRM v2.1, including:

- Enforced Advertising to ensure the full service provider control over downloaded and streamed contents.
- Metering extension for advertising. By means of metering on advertising, actual advertising information can be provided to Rights Issuers.
- Security extension for protection on multicast streaming.
- Support for multiple media formats. DRM2.2 supports different media types such as video, audio, Games/ Executables, Widget, etc. On the base of DCF2.1, in addition to the DCF and PDCF, the protection for MPEG-2 TS (Transport Stream) as MPEG2DCF is supported.

The DRM 2.2 features have minimum impact on the DRM 2.1 architecture and are defined in a DRM 2.1 backward compatible manner. The DCF v2.2 specification extends DCF 2.1 so as to support above features.

5. DRM Content Format

There are three DRM Content Format profiles:

- DCF: The first profile is intended to package and protect Discrete Media (i.e. ring tones, applications, images, etc.) The Discrete Media profile allows you to wrap any content in an envelope (DCF). That content is then encrypted as a single object agnostic of the contents internal structure and layout. This specification defines the Discrete Media format based on the types of the ISO base media file format [ISO14496-12], instead of WSP types [WSP] used in Version 1 [DRMCF-v1]. By using the ISO principles, the DCF format maintains the extensible nature of the ISO format, while keeping overhead minimal. A Device defined in [DRM-v2.2] MUST support the DCF format as defined in this specification as well as be backwards compatible with the DCF format defined in [DRMCF-v2.1]. In addition, the version 1 DCF as defined in [DRMCF-v1] MAY be supported.

- PDCF: The second profile is optimised to protect Continuous Media (e.g. Audio and Video.) Continuous media is protected in a separate profile because it is packetised and thus the profile is called the Packetised DCF (PDCF). Applications that read and parse Continuous Media are meant to work on the file on a packet-by-packet basis. To facilitate the playback of protected Continuous Media, the storage format needs to be structured in such a way that the packets are individually protected. This structurally aware packetisation is also required in order to stream Continuous Media. An OMA DRM compliant streaming server MUST be able to understand the Content Format's structure in order to break the content into headers and packets that can be delivered to a client that understands the Content Format. A Device defined in [DRM-v2.2] MAY support the PDCF format as defined in this specification; if the PDCF format is supported then the Device MUST be backwards compatible with the PDCF format defined in [DRMCF-v2.1].

- MPEG2DCF: The third profile is intended to protect MPEG-2 TS (Transport Stream). The stream format of MPEG-2 TS is defined in [ISO/IEC 13818-1]. Based on the [ISO/IEC 13818-1], the MPEG2DCF profile defines the DRM protection that enables Devices to request Rights Objects for access of broadcast content contained in MPEG-2 TS.A Device defined in [DRM-v2.2] MAY support the MPEG2DCF format as defined in this specification.

For the application of Enforced Advertising, the advertisement content SHALL be encapsulated with the normal DRM content format defined in this specification. Meanwhile, the advertisement content MAY be encapsulated and delivered either together with the normal content or separately.

5.1 ISO Base Media File Format

The Discrete Media profile (DCF) is an object-structured file as defined in section 4 of the ISO Base Media File Format specification [ISO14496-12], but it does not include all the media-related structures due to its simplified, media agnostic design. The actual data structures and conformance to the profile is defined in this specification. If a DCF includes data structures or functionalities not conforming to this specification, a compliant file parser MAY ignore these.

The Continuous Media profile (PDCF) is fully compliant with the ISO base media file format, but this specification adds support for OMA DRM 2.2 key management on top of existing ISO derived file formats supporting encrypted media content. By default, this specification addresses the DCF format, with an additional indication if a specified data structure is also used in the PDCF format.

5.1.1 ISO File structure (INFORMATIVE)

This section is informative and is based on the ISO Base Media File Format specification [ISO14496-12].

The ISO base media file format is structured around an object-oriented design of boxes. A basic box has two mandatory fields, *size* and *type*. The *type* identifier is used to dynamically bind a box to a statically defined type and the *size* is an offset from start to the end of the box. A Box type identifier is a *Unique Identifier Number.* List of reserved numbers can be found in B.3. The identifier is constructed from four bytes, each representing a human-readable character, thus the name *Four Character Code* (4CC).

The ISO base format uses a language called Syntax Description Language (SDL) for defining data structures. SDL has similarities with some programming languages and supports object orientation. The box class is the superclass for all structures containing data in the file format.

A basic box is defined as:

```
aligned(8) class Box (unsigned int(32) boxtype, optional unsigned int(8)[16] extended_type) {
      unsigned int(32) size;
      unsigned int(32) type = boxtype;
      if (size==1) {
             unsigned int(64) largesize;
      } else if (size==0) {
             // box extends to end of file
      }
      if (boxtype==‘uuid’) {
             unsigned int(8)[16] usertype = extended_type;
      }
}
```

Box alignment is by default to the next byte boundary in the end of the box. Extra padding should not be needed as all data types in e.g. the DCF are terminated on byte boundaries.

Since one of the design goals for the DCF is extensibility, it is important to carry version information with each data type. The ISO specification has a predefined type to support this, the FullBox, which is derived from the simple Box base class.

```
aligned(8) class FullBox(unsigned int(32) type, unsigned int(8) v, bit(24) f) extends Box(type) {
       unsigned int(8) version = v;
       bit(24) flags = f;
}
```

Extending a parent class has similar semantics as in many programming languages; the parent class data members precede the child class definitions. A representation of the FullBox above is:

Name	Type	Value
Size	unsigned int(32)	Offset to the end of the box
Type	unsigned int(32)	Box type 4CC
Version	unsigned int(8)	Version field
Flags	unsigned int(24)	Additional flags

The numeric fields in the ISO format are in network byte order.

5.2 Common Boxes

5.2.1 Common Headers Box

```
aligned(8) class OMADRMCommonHeaders extends FullBox('ohdr', version, 0) {
      unsigned int(8)       EncryptionMethod;     // Encryption method
      unsigned int(8)       PaddingScheme;        // Padding type
      unsigned int(64)      PlaintextLength;      // Plaintext content length in bytes
      unsigned int(16)      ContentIDLength;      // Length of ContentID field in bytes
      unsigned int(16)      RightsIssuerURLLength; // Rights Issuer URL field length in bytes
      unsigned int(16)      TextualHeadersLength;  // Length of the TextualHeaders array in bytes
      char                  ContentID[];          // Content ID string
```

```
      char                RightsIssuerURL[];    // Rights Issuer URL string
      string              TextualHeaders[];     // Additional headers as Name:Value pairs
      Box                 ExtendedHeaders[];    // Extended headers boxes
}
```

The Common Headers box defines a structure for the required headers. Their semantics are defined in the sections below. This box MUST appear in both DCF and PDCF. This box includes the mandatory headers as fixed fields and provides a mechanism to insert additional headers as arbitrary name value pairs. For application in DCF and PDCF, see sections 6.3.2 and 7.1.5.1 for details.

A Device MUST NOT modify any of the fields in the Common Headers box.

5.2.1.1 Common Headers Version

The *version* field of the `FullBox` defines which version of DRM Content Format specification was used by the author of the Content Object. The value for *version* MUST be 0 for objects conforming to this specification.

5.2.1.2 EncryptionMethod Field

The *EncryptionMethod* field defines how the encrypted content can be decrypted. Values for the field are defined in the Table 1 below.

Table 1: Algorithm-id values

Algorithm-id	Value	Semantics
NULL	0x00	No encryption for this object. NULL encrypted Content Objects may be used without acquiring a Rights Object. Value of the *PaddingScheme* field MUST be 0. In the OMADRMAUFormatBox, the values of *SelectiveEncryption* field and *IVLength* field MUST be 0.
AES_128_CBC	0x01	AES symmetric encryption as defined by NIST [AES] 128 bit keys Cipher block chaining mode (CBC) For the first block a 128-bit initialisation vector (IV) is used. For DCF files, the IV is included in the OMADRMData as a prefix of the encrypted data. For non-streamable PDCF files, the IV is included in the *IV* field of the OMADRMAUHeader and the *IVLength* field in the OMADRMAUFormatBox MUST be set to 16. Padding according to RFC 2630
AES_128_CTR	0x02	AES symmetric encryption as defined by NIST [AES] 128 bit keys Counter mode (CTR) The counter block has a length of 128 bits. For DCF files, the initial counter value is

		included in the OMADRMData as a prefix of the encrypted data. For non-streamable PDCF files, the initial counter value is included in the *IV* field of the OMADRMAUHeader and the *IVLength* field in the OMADRMAUFormatBox MUST be set to 16. For each cipherblock the counter is incremented by 1 (modulo 2^{128}). No padding.
AES_128_BYTE_CTR	0x03	AES symmetric encryption as defined by NIST [AES] and [XBS-v1.0]. 128 bit keys Counter mode (CTR) For DCF files, the initial counter value is included in the OMADRMData as a prefix of the encrypted data, and it MUST have a length of 16 bits. For non-streamable PDCF files, the initial counter value of minimal 8 bits and maximal 64 bits is included in the *IV* field of the OMADRMAUHeader. The *IVLength* field in the OMADRMAUFormatBox MUST contain the length of the initial counter value in bytes, e.g. for a 64 bit initial counter the value in *IVLength* equals 8. For this encryption method, the ExtendedHeaders field in the OMADRMCommonHeaders box MUST contain one instance of the OMADRMSalt box (see section 7.1.5.2). For each **byte of ciphertext** the counter is incremented by 1. No padding.

Rights Issuers should take care in using NULL *EncryptionMethod* because, given a null-encrypted Media Object within a DCF or PDCF, the following statements hold true:

- Null-encrypted Media Objects do not have any Confidentiality protection.
- Null-encrypted Media Objects can always be used without an associated Rights Object.
- Null-encrypted Media Objects may not have any integrity protection.

5.2.1.3 PaddingScheme Field

The *PaddingScheme* parameter defines how the last block of ciphertext is padded. Values of the *PaddingScheme* field are defined in the Table 2 below:

Table 2: PaddingScheme values

Padding-Scheme	Value	Semantics
None	0x00	No padding (e.g. when using NULL or CTR algorithm).
RFC_2630	0x01	Padding according to RFC 2630.

5.2.1.4 PlaintextLength Field

The *PlaintextLength* field defines the length of the original plaintext. In the case of DCF and if the content is encrypted, it MUST have a PlaintextLength value set. If the extracted content length does not match the PlaintextLength field value, it is an error and the Content Object MUST be discarded. In a progressive download scenario, the DRM Agent can verify the PlaintextLength only after the complete Content Object has been received and possibly after content use has started.

In the case of PDCF the PlaintextLength MUST be zero. DRM Agents should ignore the PlaintextLength for PDCF files; instead they should use the relevant ISO Base Media boxes to identify track properties.

5.2.1.5 ContentIDLength Field

The *ContentIDLength* field defines the number of bytes occupied by the *ContentID* field. The value MUST be greater than zero. A Device MUST support ContentIDs of at least 256 bytes. For best interoperability, content author should not use a ContentID larger than 256 bytes.

5.2.1.6 RightsIssuerURLLength Field

The *RightsIssuerURLLength* field indicates the number of bytes occupied by the *RightsIssuerURL* field. A Device MUST support RightsIssuerURLs of at least 256 bytes. For best interoperability, content author should not use a RightsIssuerURL larger than 256 bytes.

5.2.1.7 TextualHeadersLength Field

The *TextualHeadersLength* field indicates the number of bytes occupied by the *TextualHeaders* field. Although it is possible with this version of the parent box to implicitly determine the TextualHeaders field length from the box length, this might not be the case in future versions. Thus, conforming tools MUST use the TextualHeadersLength field. A Device MUST support textual headers of at least 2048 bytes total length.

5.2.1.8 ContentID Field

The *ContentID* field MUST contain a globally unique identifier for this Content Object. Note that even if two or more (P)DCFs contain the same Content Object, the Content Objects will each have a different (and globally unique) ContentID. The value MUST be encoded using US-ASCII encoding.

The value MUST be a unique URI according to [RFC2396]. The use of globally unique *ContentID*'s is required for OMA DRM and it is the responsibility of the content author to guarantee the uniqueness of the *ContentID* within their own namespace.

If the Content Object is referenced from a DRM Rights Object, the value of the *ContentID* field MUST match the value of the referencing element of the Rights Object as defined in [DRMREL-v2.2]. The ContentID MUST be in the 'cid-url' format of [RFC2392].

In the case of multi-part DCFs and multi-track PDCFs the first ContentId in the (P)DCF MUST represent the entire file. The entire file is referenced in the [DRM-v2.2] ROAP protocols for associating TransactionIDs and DCF Hash values with particular files. The first track in a multi-track PDCF is the protected track with the lowest integer "`track_ID`" in the PDCF `Track Header Box ('tkhd')`.

5.2.1.9 RightsIssuerURL Field

The *RightsIssuerURL* field defines the Rights Issuer URL. The Rights Issuer URL MAY be used by the consuming Device to obtain Rights for this DRM Content. The mechanism is defined in OMA DRM specification [DRM-v2.2]. The value of the

RightsIssuerURL field MUST be encoded using US-ASCII encoding. The length of this field is indicated by the *RightsIssuerURLLength* field.

The value of the *RightsIssuerURL* MUST be a URL according to [RFC2396], and MUST be an absolute identifier. The *RightsIssuerURL* MAY be empty e.g. if the Content Object is not encrypted.

5.2.2 Textual Headers

The *TextualHeaders* field MAY contain additional information about the content.

Textual headers are represented by name value pairs, where name and value are separated with a colon ':' and the pair is terminated with a NULL ('\0') character. A header (name value pair) MUST NOT include leading or trailing whitespace (such as \r\n). Further, a header name MUST NOT include a colon (':') character, as the first instance or the character will stop scanning for the header name. Header value MAY include colon characters as the value is always assumed to continue after the first colon until a NULL character is reached.

The next header name MUST begin immediately after the terminating NULL character of the previous header, if TextualHeadersLength is greater than the current scanning position. All headers MUST have a value, i.e. an empty value is not permitted.

The textual headers field continues until the TextualHeadersLength offset or the end of the box is reached. The TextualHeadersLength field MUST be used to determine the TextualHeaders field length.

An example representation of the textual headers:

```
Silent:on-demand;http://myissuer.com/silent?cid=428\0Preview:instant;cid:429@myissuer.com\0
```

Each supported header is defined using augmented Backus-Naur Form (BNF) [RFC2234]. The textual headers are encoded using UTF-8 encoding. Ordering of headers is significant, and the headers MUST be in the order of priority, from highest to lowest. This means that e.g. if the textual headers include both Silent and Preview headers, whichever appears first in the field is considered to have priority over the second.

5.2.2.1 Silent header

The *Silent* header is an indication to the client that the Rights Object for this DRM Content can be obtained silently from the Rights Issuer, without user interaction for payments, etc.

```
Silent = "Silent" ":" silent-method ";" parameter
silent-method = token
parameter = silent-rights-url
silent-rights-url = token
```

silent-method	Semantics
"on-demand"	Rights should be acquired silently, on demand when the user chooses to play the content.
"in-advance"	Rights should be acquired in advance, at the earliest opportunity.

The parameter `silent-rights-url` MUST be a URL according to [RFC2396] and a successful request to the URL MUST return a ROAP Trigger, a Download Descriptor or a bundled Download Descriptor and ROAP Trigger as defined in [DRM-v2.2]. If `silent-rights-url` is a HTTP URL and the request fails with error code `404 Not Found` [RFC2616], the Device SHOULD NOT make further requests to the URL. If the request fails with some other error, the Device MAY retry the request at a later time.

The parameter `silent-rights-url` MUST be specified on the `Silent` header. The Device MUST use this `silent-rights-url` to obtain rights silently and automatically according to [DRM-v2.2].

5.2.2.2 Preview header

The *Preview* header contains an indication to the client that it is possible to provide a preview for this DRM Content.

If the `preview-method` is "`instant`", then the specific media element to be used for preview MUST be indicated using the `preview-element-uri` parameter. In addition, this media element MUST be NULL-encrypted, and as such, MUST have an *`EncryptionMethod`* header with the `algorithm-id` parameter set to `NULL`.

```
Preview = "Preview" ":" preview-method (";" parameter )
preview-method = token
parameter =  preview-element-uri | preview-rights-url
preview-element-uri = token
preview-rights-url = token
```

Preview-method	Semantics
"`instant`"	This indicates that one of the elements within this file can be used for preview. If `instant` method is specified, then `preview-element-uri` MUST be specified.
"`preview-rights`"	This indicates that a preview Rights Object can be obtained by requesting it silently from the Rights Issuer, without user interaction If `preview-rights` method is specified, then `preview-rights-url` MUST be specified.

The parameter `preview-element-uri` MUST be a unique identifier and a URI according to [RFC2396]. And, it MUST resolve to an element present within the same file.

The parameter `preview-rights-url` MUST be a URL according to [RFC2396] and a successful request to the URL MUST return a ROAP Trigger, a Download Descriptor or a bundled Download Descriptor and ROAP Trigger as defined in [DRM-v2.2]. If `preview-rights-url` is a HTTP URL and the request fails with error code `404 Not Found` [RFC2616], the Device SHOULD NOT make further requests to the URL. If the request fails with some other error, the Device MAY retry the request at a later time.

If the `preview-method` is indicated as "instant", the preview element can be used freely with unlimited use, without acquiring any Rights Objects.

If the `preview-method` is "`preview-rights`", then the `preview-rights-url` MUST be indicated as a parameter. When the client connects to the Rights Issuer with this URL, this MUST NOT result in any re-direction.

5.2.2.3 ContentURL header

The *ContentURL* header is used to indicate a location for acquiring the DCF or PDCF. This MAY be used to e.g. download an alternative version of the file if a Device does not support the content types in the current file, such as resolution or codec. The consuming Device MAY provide the option to forward the ContentURL to other users as an alternative form of superdistribution. The mechanism is defined in OMA DRM specification [DRM-v2.2].

```
ContentURL = "ContentURL" ":" content-url
content-url = token
```

The `content-url` MUST be a URL according to [RFC2396] and MUST be an absolute identifier. The Device MAY access the ContentURL from the DCF and use it to establish e.g. a browsing session without acquiring a Rights Object for the DRM Content.

5.2.2.4 ContentVersion header

The *ContentVersion* header defines the version of the content. This header MAY be used to uniquely identify the incarnation of this DRM Content Object.

```
ContentVersion = "ContentVersion" ":" original-content-identifier ":" version-identifier
```

```
original-content-identifier = token
version-identifier = *digit
```

Where `original-content-identifier` MUST be a matching string for all versions of the same Content and `version-identifier` MUST be a number in range 0..65535, incremented by each version.

5.2.2.5 Content-Location header

The *Content-Location* header MAY be used to indicate a relative location for the Content Object. This MAY be used for e.g. referencing purposes within the DCF file or determining a meaningful file name when exporting the Content Object.

```
ContentLocation = "Content-Location" ":" content-uri
content-uri = token
```

The `content-uri` MUST be a file name, relative to the location of the DCF file.

5.2.2.6 Custom headers

Content author MAY insert additional Custom headers to the *TextualHeaders* field. Custom headers MUST follow the generic syntax defined below, encoded using UTF-8 encoding.

```
OtherHeader = Header-name ":" Header-value
Header-name = token
Header-value = token
```

Consuming Devices MUST ignore the headers that they do not recognize.

5.2.2.7 ProfileName header

The *ProfileName* header contains a profile name for this DRM Content. The purpose of a profile name is to define in detail what resources are needed to render this content (e.g. codec, profile and level). Note that this specification does not specify actual profile names.

```
ProfileName = "ProfileName" ":" profile-name-uri
profile-name-uri = token
```

The `profile-name-uri` MUST be a unique identifier and a URI according to [RFC2396]. In addition, it MUST resolve to a profile name specified by some standardisation body (e.g. "//www.dlna.org/AAC_ISO_320").

Content Authors SHOULD insert a ProfileNames box, containing one or more Profile Names in order to provide a simple way whether this content can be rendered on a specific Device or not.

5.2.3 Extended Headers

The *ExtendedHeaders* field MAY include zero or more nested boxes that add functionalities to the common headers. The ExtendedHeaders field continues until the end of the parent box is reached.

5.2.3.1 Group ID

The *ExtendedHeaders* field MAY include one instance of the *OMADRMGroupID* Box:

```
aligned (8) class OMADRMGroupID extends FullBox('grpi', version, 0) {
      unsigned int(16) GroupIDLength;             // length of the Group ID URI
      unsigned int(8) GKEncryptionMethod;         // Group Key encryption algorithm
      unsigned int(16) GKLength;                  // length of the encrypted Group Key
      char GroupID[GroupIDLength];                // Group ID URI
      byte GroupKey[GKLength];                    // Encrypted Group Key and encryption information
}
```

The *GroupID* value identifies this DCF as part of a group of DCF's whose Rights can be defined in a common group Rights Object instead of (or in addition to) in separate content-specific Rights Objects. The value of *GroupID* MUST be a URI according to [RFC2396] and MUST contain a globally unique identifier. Further the GroupID MUST use the URL format of [RFC2392] except the scheme name must be "gid:" The value MUST be encoded using US-ASCII encoding.

Generally each content item in a group will be encrypted with a different content item encryption key. A single additional key (used for the whole group) is used to encrypt each content item encryption key for storage in the GroupKey field. This single key is the value of the CEK in an associated group RO. Note that since the Group ID box is part of the OMA DRM container box, it is possible for different content items in a multipart DCF to belong to different groups. The GKEncryptionMethod field defines the algorithm used to encrypt the content item encryption keys, as defined in Section 5.2.1.2, and it defines the structure of the GroupKey field that can contain, next to the actual encrypted content item encryption key (refered to in Section 5.2.1.2 as 'ciphertext'), additional information such as initialisation vector or initial counter value. The NULL *EncryptionMethod* MUST NOT be used as a GKEncryptionMethod.

Table 3: Group ID box fields

Field name	Type	Purpose
GroupIDLength	unsigned int(16)	Length of the Group ID URI field
GKEncryptionMethod	unsigned int(8)	Group Key encryption algorithm
GKLength	unsigned int(16)	Length of the GroupKey field
GroupID	char[]	Group ID URI
GroupKey	byte[EncryptedGKLength]	Encrypted Group Key and additional encryption information such as initialisation vector, counter values, padding as defined in Section 5.2.1.2

5.2.3.2 Old Content ID

```
aligned (8) class OldContentID extends FullBox('grpi', version, 0) {
      unsigned int(16)      ContentIDLength;      // Length of ContentID field to be updated in bytes
      char                  ContentID[];          // Content ID string to be updated
}
```

If the DCF is used for the Dynamic Advertisement Update (see [DRM-v2.2]), the OMA DRM Container MUST carry *OldContentID* in its extended headers.

The *ContentIDLength* field defines the number of bytes occupied by the *ContentID* field as defined in Section 5.2.1.5.

The *ContentID* field MUST contain a globally unique identifier for this Content Object as defined in Section 5.2.1.8.

5.2.4 Mutable DRM Information Box

The `MutableDRMInformation` box MAY appear in both DCF and non-streamable PDCF. `MutableDRMInformation` is not applicable to streamable PDCF. In the OMA DRM system, the `MutableDRMInformation` box is used to include information editable by the Device, and thus is not protected for integrity. A Device MUST ignore the `MutableDRMInformation` box when calculating the DCF hash.

The `MutableDRMInformation` box MUST be located at the top level of the box hierarchy and there MUST NOT be more than one instance of the box per DCF or PDCF. The `MutableDRMInformation` box MAY include free space boxes as defined in ISO base media file format [ISO14496-12] to pre-allocate space for editing. A `MutableDRMInformation` box MUST NOT appear in the beginning of the file, but MAY appear after the last `OMADRMContainer` (see 6.3.1) in DCF and after the movie box in PDCF. Having the `MutableDRMInformation` box as the last box in the file is RECOMMENDED for DCF.

For PDCF the location of the `MutableDRMInformation` should be carefully considered by the Content Issuer. Generally it is preferable to place the `MutableDRMInformation` directly after the Movie Box, this enables Transaction Tracking and Rights Object delivery during Progressive Download. However, if the `MutableDRMInformation` box is after the movie box it will be difficult for the client to insert new Rights Objects into the Rights Object box because if the size of the MutableDRMInformation box changes the Device must also update the Chunk Offset Box ('stco') in the PDCF headers. Therefore Content Issuers are recommended to also include a Free Space Box if the `MutableDRMInformation` is placed before the Media Data.

A Device MAY modify, extend, truncate, delete or add the `MutableDRMInformation` box. The contents of the box MUST be interpreted as an array of Boxes, continuing until the end of the parent box.

```
 aligned(8) MutableDRMInformation extends Box('mdri') {
      Box     data[];               // array of any boxes and free space
}
```

5.2.4.1 Transaction Tracking Box

The OMA DRM Transaction Tracking Box enables transaction tracking as defined in [DRM-v2.2] section 15.3. The `OMADRMTransactionTracking` box MUST include a single *TransactionID* value as defined below. It MAY appear in both DCF and PDCF.

```
aligned(8) class OMADRMTransactionTracking extends FullBox('odtt', 0, 0) {
      char    TransactionID[16];     // value to enable transaction tracking
}
```

Table 4: OMA DRM transaction tracking header field

Field name	Type	Purpose
TransactionID	char[16]	TransactionID of the DCF or PDCF respectively

The Rights Issuer MAY provide any value as a TransactionID to the DRM Agent during the Rights acquisition process and the TransactionID included in the DRM Container may be changed by the DRM Agent as defined in [DRM-v2.2]. When packaging content, the TransactionID MAY be set to an arbitrary value.

As per [DRM-v2.2] section 15.3 the DRM Agent does not need to generate the `OMADRMTransactionTracking` box, nor does the DRM Agent ever need to modify the size of the box.

5.2.4.2 Rights Object Box

The rights object box MAY be used to insert a Protected Rights Object, defined in [DRM-v2.2] section 5.3.9, into a DCF or PDCF. A `MutableDRMInformation` box MAY include zero or more Rights Object boxes. The Rights Object is treated as binary data and a Device MAY add or delete Rights Object boxes in the `MutableDRMInformation` box.

```
aligned(8) class OMADRMRightsObject extends FullBox('odrb', 0, 0) {
      byte    Data[];               // binary Rights Object
}
```

Table 5: OMA DRM Rights Object box fields

Field name	Type	Purpose
Data	byte[]	A Rights Object as binary data

5.2.4.3 User-Data Box

A *MutableDRMInformation* box MAY include one or more User-Data boxes ('udta'), as defined in 6.3.2.3.

The insertion of a User-Data box in the *MutableDRMInformation* box allows Users and Devices to add to and edit the metadata associated with DRM Content.

The corresponding DRM Content is identified by the ContentID sub-box.

When available the Device SHALL use the metadata information stored in the User-Data box in the *MutableDRMInformation* box, instead of the equivalent metadata information stored in the User-Data box in the *OMADRMDiscreteHeaders box.*

If it is desirable to allow User editing of metadata information, it is RECOMMENDED that the appropriate free space boxes in the MutableDRMInformation box are inserted to facilitate User and Device edits.

Mutable DRM Information Box MAY include Advertisements and it can be updated by advertiser.

5.2.4.3.1 ContentID sub-box

```
aligned(8) class OMADRMContentID extends FullBox('ccid', version, 0) {
        unsigned int(16)      ContentIDLength;      // Length of ContentID field in bytes
        char                  ContentID[];          // Content ID string
}
```

The ContentID box (‘ccid’) contains the unique identifier for the Content Object the metadata are associated with.

The value of the *ContentID* MUST be the value of the ContentID stored in the Common Headers for this Content Object.

There MUST be exactly one *ContentID* sub-box per User-Data box, as the first sub-box in the container.

Table 6: ContentID box

Field name	Type	Purpose
`ContentIDLength`	`unsigned int(16)`	`Length of ContentID field in bytes`
`ContentID`	char[]	`Content ID string`

5.2.4.3.2 Other User-Data sub-boxes

The User-Data box in the *MutableDRMInformation* MAY include any User-Data sub-boxes as defined in 6.3.2.3.

5.3 DCF Hash Calculation

Content Objects MAY be protected for integrity by including a DCF hash into a Rights Object or ROAP request. Since (P)DCF MAY include structures editable by the Device, these structures are excluded from hash calculation. The DCF hash MUST be calculated from the beginning of the DCF to the end of the last `OMADRMContainer`, ignoring the `MutableDRMInformation` box. PDCF hash MUST be calculated from the beginning of the PDCF, skipping the `MutableDRMInformation` box after the movie box, or end of file in case there is no `MutableDRMInformation` box present.

6. Discrete Media Profile (DCF)

This section defines the DRM Content Format for Discrete Media.

6.1 DCF MIME Type

The MIME type for objects conforming to the format defined in this section MUST be

```
application/vnd.oma.drm.dcf
```

By default file extension SHOULD be ".odf", however, to allow for systems that use file extensions to indicate the type of content without requiring the parsing of the DCF headers a DCF MAY also have an ".o4a" or ".o4v" extension. Here an "o4a" indicates that the default media type of the Content Object(s) in the DCF is music whilst an ".o4v" indicates that the default media type is video..

- If the default media type of the Content Object(s) in the DCF is music then the DCF file extension SHOULD be ".o4a".
- If the default media type of the Content Object(s) in the DCF is video then the DCF file extension SHOULD be ".o4v"
- For all other media types the DCF file extension MUST be ".odf".

The DCF file extension MUST be either ".odf", "o4a" or "o4v". For DCFs that contain multiple OMA DRM Containers the default media type is defined to be the media type of the first OMA DRM Container as specified in section 6.4.

6.2 DCF File Format

The structure of the Discrete Media profile of DRM Content Format (DCF) MUST be according to the structure definitions below.

A DCF file MUST include at least one `OMADRMContainer` box. The `OMADRMContainer` box is a container for a single Content Object and its associated headers. It MUST appear on the top level, i.e. to conform to this specification, it MUST NOT be nested inside another data type. There MAY exist multiple `OMADRMContainer` boxes in a file, but one MUST immediately follow the file header, and they all MUST be located on the top level in the nesting structure.

The *version* indicator field in each box MUST be 0 for files conforming to this specification. All numeric fields in the format MUST be stored in network byte order.

6.2.1 OMA Constraints on ISO Format

In files conforming to this specification, box *size* MUST be greater than 1 unless otherwise specified and the *extended_type* MUST NOT be used in the mandatory boxes. Some of the mandatory boxes MUST support the 64 bit length field and for those boxes, *size* field MUST be set to 1. Also note that in some earlier ISO specifications, the term `atom` was used to describe the file format structures, but the structures specified in this specification are called `box`es in order to be consistent with current specifications.

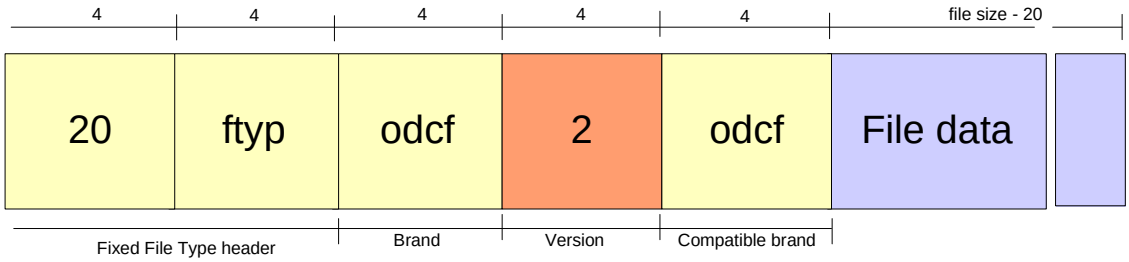
The FullBox *version* is typically started from zero (0), incremented by each revision. The *flags* field MAY be used to include additional information, but SHOULD normally be set to 0, unless otherwise specified. This specification names each supported box to indicate that a box has a defined structure and a purpose in the OMA DRM Content Format.

There are also placeholders for extensions, with only a generic box reference. These extensions may be defined later, and thus a conforming file parser SHOULD skip any extension boxes it does not understand. In addition, all of the toplevel boxes are derived from the FullBox type, which supports version information. Later specifications MAY increment the version number if changes are made to any common data structures. Later versions of the boxes defined in this specification should remain backwards compatible with the help of this version indicator. A parser conforming to this specification MAY attempt to parse a box which has a greater version number than this specification, but the conformance is limited to the current version (0) of this specification. A conforming parser MUST check the version number field.

6.2.2 File Branding

The ISO base media file format defines a File Type box for identifying the major brand of the media file along with compatible brands. Files conforming to the Discrete Media profile MUST include a File Type box with the DCF brand as the major brand number and compatible brand to make the File Type box fixed length. The DCF major brand is 32 bits (4 octets) wide with the hexadecimal value `0x6F646366` (‘`odcf`’). This MUST be followed by a four-octet minor version indicator and the DCF brand as the single compatible brand, making the file header a total of 20 octets (160 bits) from the beginning of the file. The minor version field is in network byte order. For files conforming to this version of the DCF specification the version value MUST be 2 (`0x00000002`). A conforming file parser MUST support the minor version number. It should be noted that future minor versions of the DCF file format might use more compatible brands in the File Type box, changing the file header length. The Figure 1 shows the relationship of the File Type, brand, version and rest of the file content.

Figure 1: DCF file header and body

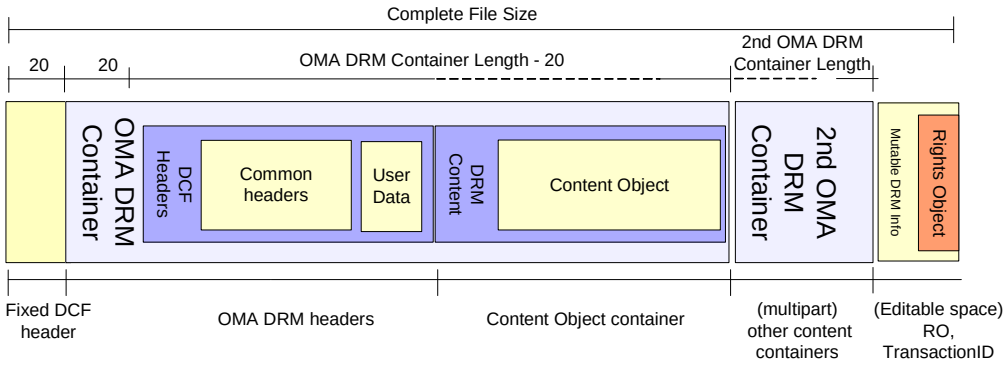


6.3 Overall structure

The high-level overview of the DCF format is depicted in the Figure 2. The mandatory parts of the format include the file header (File Type box with brand number and minor version fields), immediately followed by an OMA DRM Container box. The OMA DRM Container box MUST include a DCF headers box and a Protected Content box.

The design principles for the format include that the DCF headers box is located at a fixed offset from the beginning of the file, and thus, the OMA DRM Container box MUST be the first box after the file header of 20 octets and the DCF headers box MUST be the first box in the OMA DRM Container. Multipart DCFs support Advertisement management. So the multipart DCF MAY contain the DRM Contents with Advertisements.

Figure 2: DCF structure



The Table 7 below outlines the mandatory boxes and their order. Additional boxes MAY be added after the mandatory boxes have first appeared. Table 7 shows the nesting order of the mandatory boxes, on the left is the parent and on the right, the child. The first column indicates which fields and boxes MUST be present in DCF (marked as ‘M’) and which boxes MAY appear in the DCF (marked as ‘O’). Note that in the table, the second OMA DRM Container box MUST include all the mandatory nested boxes as well.

Table 7: Logical DCF box structure diagram

Present in DCF	Data type/value			Nesting level	Offset from beginning of file	Field purpose
M	Box('ftyp')			0	0	File header (fixed File Type box, 20 bytes)
M	Box('odrm')			0	20	OMA DRM Container box
M		Box('odhe')		1	40	Discrete Media headers box
M			Box('ohdr')	2	53 + ContentTypeLength	OMA DRM Common Headers box
O			Box('udta')	2	53 + ContentTypeLength + Box('ohdr')	ISO User Data box (optional)
M		Box('odda')		1	40 + Box('odhe')	Content Object box
O	Box('odrm')			0		If multipart DCF, additional OMA DRM Container box
O	Box('mdri')			0		Mutable DRM information box
O		Box('odtt')		1		Transaction tracking box
O		Box('odrb')		1		Rights Object container box
O		Box('udta')		1		ISO User Data box (optional)
M			Box('ccid')	2		ContentID reference for the User-Data
O		Box('skip')		1		Additional free space

6.3.1 OMA DRM Container Box

```
aligned(8) class OMADRMContainer extends FullBox('odrm', version, 0) {
      OMADRMDiscreteHeaders  ContentHeaders;       // Headers for Discrete Media DCF
      OMADRMContentObject    DRMContent;           // Actual encrypted content
      Box                    Extensions[];         // Extensions, to the end of the box
}
```

The `OMADRMContainer` box MUST include a single `OMADRMDiscreteHeaders` box and a single `OMADRMContentObject` box, followed by optional extensions. The `Extensions` inside the `OMADRMContainer` box are defined by OMA. The OMA DRM Container box MUST support 64 bit length attributes, i.e. the *size* attribute MUST be set to 1, and *largesize* MUST be used for determining the box size.

6.3.2 Discrete Media Headers Box

```
aligned(8) class OMADRMDiscreteHeaders extends FullBox('odhe', version, flags) {
      unsigned int(8)       ContentTypeLength;     // Content Type Length
```

```
      char                  ContentType[];        // Content Type String
      OMADRMCommonHeaders   CommonHeaders;        // Common headers (same as with PDCF)
      if(flags & 0x000001) {
             UserDataBox           UserData;      // ISO User Data Box (optional)
      }
}
```

The Discrete Media headers box includes fields specific to the DCF format and the Common Headers box, followed by an optional user-data box. There MUST be exactly one `OMADRMDiscreteHeaders` box in a single OMA DRM Container box, as the first box in the container.

The *ContentType* field indicates the actual media type contained in the OMA DRM container. There MUST be exactly one `OMADRMCommonHeaders` (see section 5.2.1 for details) box per a single `OMADRMDiscreteHeaders` box.

Table 8: OMA DRM Discrete Media header fields

Field name	Type	Purpose
ContentTypeLength	Unsigned int(8)	Length of the ContentType field
ContentType	ContentTypeLength octets	The MIME media type of the plaintext data encoded as US-ASCII
CommonHeaders	OMADRMCommonHeaders	OMA DRM Common Headers box as in 5.2.1
UserData	UserDataBox	User Data as defined in 6.3.2.3 (OPTIONAL)

6.3.2.1 ContentType

The *ContentType* field MUST indicate the original MIME media type of the Content Object i.e. what content type the result of a successful extraction of the `OMADRMContent` box represents. The *ContentType* field is encoded using US-ASCII encoding and MUST NOT include a NULL character.

6.3.2.2 CommonHeaders

The *CommonHeaders* field MUST be the same box as defined in 5.2.1.

6.3.2.3 User-Data

A user-data box ('udta'), as defined in [ISO14496-12], MAY be present in the discrete headers box. When a DCF includes the `UserData`Box, it MUST be added immediately after the `OMADRMCommonHeaders` box. The presence of the user-data box MUST be indicated with the flag 0x000001 in the containing box header. The user-data box is a container box for informative user data. This user information is formatted as a set of sub-boxes with specific box types that more precisely define their usage. Each of the sub-boxes MAY be included only once unless otherwise noted.

Some of these sub-boxes contain text information, which is metadata, as defined in [TS26.244]. This specification supports a superset of the sub-boxes defined in [TS26.244].

A Device MUST NOT modify any of the sub-boxes in the User-Data box in the *OMADRMDiscreteHeaders*.

The User-Data in the *OMADRMDiscreteHeaders* MAY be protected for integrity by using a DCF hash.

6.3.2.3.1 Basic User-Data sub-boxes

The User-Data box in the *OMADRMDiscreteHeaders* MAY include any User-Data sub-boxes as defined in 3GPP [TS26.244].

The following sub-boxes are currently specified:

- titl – title for the media (see [TS26.244] table 8.1)
- dscp – caption or description for the media (see [TS26.244] table 8.2)
- cprt – notice about organisation holding copyright for the media file (see [TS26.244] table 8.3)
- perf – performer or artist (see [TS26.244] table 8.4)

- auth – author of the media (see [TS26.244] table 8.5)
- gnre – genre (category and style) of the media (see [TS26.244] table 8.6)
- rtng – media rating (see [TS26.244] table 8.7)
- clsf – classification of the media (see [TS26.244] table 8.8)
- kywd – media keywords (see [TS26.244] table 8.9)
- loci – location information (see [TS26.244] table 8.10)
- albm – album title and track number for the media (see [TS26.244] table 8.11)
- yrrc – recording year for the media (see [TS26.244] table 8.12)

Each of those sub-boxes MAY be included zero, one or more time as noted in [TS26.244].

A Device MUST support UTF-8 encoded text and MAY support UTF-16 encoded text in each of these sub-boxes.

6.3.2.3.2 IconURI

```
aligned(8) class OMADRMIconURI extends FullBox('icnu', version, 0) {
      char                  IconURI[];            // Icon URI
}
```

The IconURI box (‘icnu’) contains a URI where an appropriate icon for this content may be retrieved from. The Device MAY request the object at this URI, and if an appropriate content is returned, use this as an icon associated with the content to the user.

The value of the *IconURI* MUST be a URI according to [RFC2396]. It is a string encoded using UTF-8 characters, continuing until the end of the box is reached.

If the DCF is a Multipart DCF, a *IconURI* MAY be a CID reference [RFC2557] within the current file. In this case, the referenced Content Object MUST be NULL-encrypted.

Table 9: IconURI box

Field name	Type	Purpose
IconURI	char[]	URI for an Icon for the content.

6.3.2.3.3 InfoURL

```
aligned(8) class OMADRMInfoURL extends FullBox('infu', version, 0) {
      char                  InfoURL[];            // Info URL
}
```

The InfoURL box (‘infu’) contains a URL where additional information can be found regarding the Content Object. The Device MAY obtain this information prior to using the *RightsIssuerURL* field or after the Rights Object has been obtained.

The value of the *InfoURL* MUST be a URL according to [RFC2396] and MUST be an absolute identifier. It is a string encoded using UTF-8 characters, continuing until the end of the box is reached.

Table 10: InfoURL box

Field name	Type	Purpose
InfoURL	char[]	Location of additional information for the content.

6.3.2.3.4 CoverURI

```
•     aligned(8) class OMADRMCoverURI extends FullBox('cvru', version, 0) {
•           char                  CoverURI[];           // Cover URI
•     }
```

The CoverURI box (‘cvru') contains a URI where an appropriate Artwork picture (ex: CD Album Cover, DVD front Cover, …) for this content MAY be retrieved from. The Device MAY request the object at this URI, and if an appropriate content is returned, use this as an Artwork associated with the content to the User.

The value of the CoverURI MUST be a URI according to [RFC2396]. It is a string encoded using UTF-8 characters, continuing until the end of the box is reached.

If the DCF is a Multipart DCF, a CoverURI MAY be a CID reference [RFC2557] within the current file.
The referenced Content Object MAY be DRM protected.

Table 11: CoverURI box

Field name	Type	Purpose
CoverURI	char[]	URI for an ArtWork picture for the content.

6.3.2.3.5 LyricsURI

```
aligned(8) class OMADRM LyricsURI extends FullBox('lrcu', version, 0) {
        char                  LyricsURI [];         // Lyrics URI
}
```

The LyricsURI box (‘lrcu') contains a URI where an appropriate Lyrics text for this content MAY be retrieved from. The Device MAY request the object at this URI, and if an appropriate content is returned, use this as a Lyrics associated with the content to the User.

The value of the LyricsURI MUST be a URI according to [RFC2396]. It is a string encoded using UTF-8 characters, continuing until the end of the box is reached.

If the DCF is a Multipart DCF, the value of the LyricsURI MAY be a CID reference [RFC2557] within the current file.
The referenced Content Object MAY be DRM protected.

Table 12: LyricsURI box

Field name	Type	Purpose
LyricsURI	char[]	URI for Lyrics text for the content.

6.3.2.3.6 Custom User-Data sub-boxes

Content author MAY insert additional Custom sub-boxes to the User-Data.

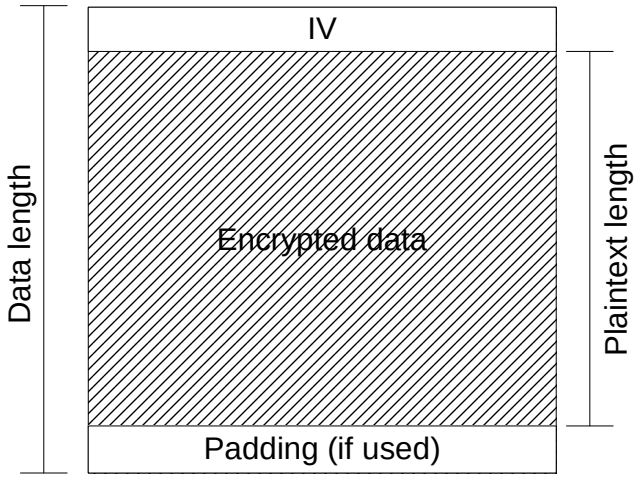
Consuming Devices MUST ignore the User-Data sub-boxes that they do not recognize.

6.3.3 Content Object Box

```
aligned(8) class OMADRMContentObject extends FullBox('odda', version, 0) {
      unsigned int(64) OMADRMDataLength;    // Length of the encrypted data plus length of the
                                               IV/counter plus length of optional padding
      byte    OMADRMData[];                // Encrypted data plus IV/counter plus optional padding
}
```

The Content Object box MUST include only the data length field and data bytes for a single Content Object. Later revisions of this box may include additional fields, so conforming implementations MUST use the *OMADRMDataLength* field to indicate/determine the amount of actual data bytes. The data length includes the Initialisation Vector in the beginning of the encrypted data, as depicted in Figure 3.

Figure 3: Data Length and IV



The Content Object box MUST support the 64 bit size field and thus *size* MUST be set to 1 and *largesize* MUST be used for determining actual box size. The *OMADRMDataLength* field MAY indicate a length of zero, and the Device MAY try to acquire the actual Content Object by using e.g. the *ContentURL*, if provided.

Table 13: Content Object box

Field name	Type	Purpose
OMADRMDataLength	Unsigned int(64)	Length of the OMADRMData field, in octets
OMADRMData	byte []	Content bytes, as specified by the OMADRMDiscreteHeaders box

The Content Object Box MAY include Advertisements also it can be updated by advertiser.

6.3.4 Extended Boxes

Any additional boxes contained in a single OMA DRM container box have not been defined in this specification. A Content Issuer MAY place additional boxes into the *Extensions* but Devices MAY ignore these.

6.4 Multiple OMA DRM Containers

A DCF MAY include more than one OMA DRM Container. Each of these containers MUST conform to the definition of the OMA DRM Container, and MUST be placed sequentially on the top level (i.e. nesting them is not allowed). The media type of Content Object in each these containers MAY BE different. However, the media type of the first OMA DRM Container is considered to be the default media type of the DCF's content.

Each OMA DRM Container MUST have a unique *ContentID* in its headers. This kind of a DCF with multiple Content containers is called a Multipart DCF.

Note that a Multipart DCF is different from a DCF including a Composite Object. A Composite Object (such as MIME multipart, ZIP and so on) is included in a single OMA DRM Container and has only one set of OMA DRM headers associated with it, whereas Multipart DCFs contain multiple OMA DRM Containers each including separate headers associated with the contained content. Multipart DCFs support the association of different rights with individual Media Objects.

6.4.1 Referencing Multipart Objects

As each object in the Multipart DCF has its own ContentID and MAY have a Content-Location header, the CID mechanism from [RFC2557] or the Content-Location mechanism from [RFC2616] MUST be used for referencing objects within the Multipart DCF. The reference MAY then be used in e.g. multimedia presentations to include objects from within the Multipart DCF. Individual Content Objects cannot be referenced from e.g. presentations outside the DCF file.

The ContentID is considered to be internal for the DRM Content Format and DRM Agent, and ContentIDs are referenced from outside the DRM Content Format only to associate it with a Rights Object. Transport protocols MUST define their own mechanisms how to reference to a DRM Content Format file.

6.5 Additional Extensions

Additional extension boxes MAY be added after the first OMA DRM Container. A conforming file parser, which does not recognize the additional boxes, MUST ignore them. However, any extensions MUST be designed in a way that the mandatory parts of this specification are always included and the file remains interoperable with conforming implementations.

7. Continuous Media Profile (PDCF)

The Continuous (Packetised) Media profile is targeted for media content like audio and video. Audio and video files MAY be included in a DCF format, but since the PDCF format has been specifically designed for Continuous Media, it provides additional advantages for those media types.

The PDCF format is an instance of the ISO Base Media File Format [ISO14496-12] that supports encrypted media tracks, which MUST use OMA DRM for key management and MUST include the OMA DRM data structures defined in this specification. Examples of ISO Base Media File Format instantations are the 3GP format [TS26.244] and 3G2 format [C.S0050].

The PDCF format MAY be used for downloaded content or for hosting streamable content. OMA DRM specifies common data structures for file formats and additional information on top of streaming services. The OMA DRM 2.2 specifications define key management functionality supporting Continuous Media but services can optimise the protocols and codecs in their architecture. Supporting the PDCF format is OPTIONAL for a Device.

7.1 PDCF File format

7.1.1 File branding

The ISO base media file format defines a File Type box for identifying the major brand of the media file along with compatible brands. Files conforming to the Continuous Media profile MUST include a File Type box with the PDCF brand as compatible brand. The PDCF brand is not recommended to be used as a major brand. Note: the major brand for the PDCF should be the same as for the unprotected file. Unlike the File Type box defined for DCF, the File Type box in PDCF does not have a fixed length of 20 bytes.

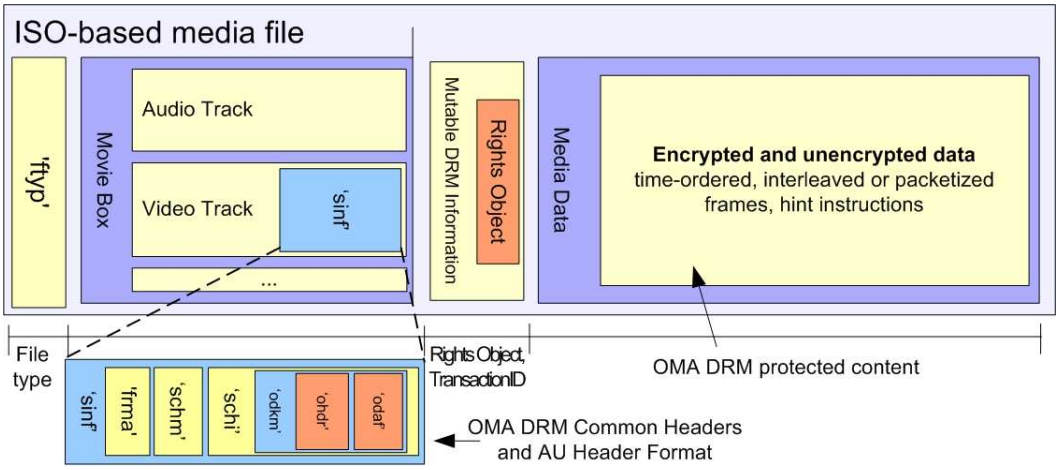
The PDCF brand is 32 bits (4 octets) wide and MUST have the hexadecimal value `0x6F706632` (‘`opf2`’). If the PDCF brand is used as major brand, this is followed by the four-octet minor version indicator. To signalize that the PDCF is conform to this specification, the minor_version MUST have the value 0 (`0x00000000`). However, higher minor_versions MAY be used in future PDCF specifications to signalize compatible extensions.

7.1.2 Overall structure

This specification defines the OMA DRM key management part of the PDCF format. In the `ProtectionSchemeInfoBox` (defined in [ISO14496-12]), there is space for a “black box” (SchemeInformationBox) describing the key management governing access to the encrypted media content. In a PDCF file, this box MUST be the `OMADRMKMSBox`.

The basic PDCF file format data structures are defined by the corresponding base file format specification, and this specification only adds OMA DRM specific structures and parameters. Other DRM mechanisms MAY be used in those file formats supporting encrypted media tracks, but not in PDCF files, as explained in this specification.

Figure 4: Example PDCF Structure



The Figure 4 illustrates how protection information can be stored in a PDCF. It is an example where a video track is protected by placing a `ProtectionSchemeInfoBox ('sinf')` into a *Protected Sample Entry* within the track's `SampleDescriptionBox ('stsd')`. The `ProtectionSchemeInfoBox` specifies the OMA DRM identifier as the key management system. Each track in a PDCF can be protected by similarly placing a `ProtectionSchemeInfoBox` into the track.

This version of the specification does not support protection of Meta Data Items in PDCF files; therefore a `ProtectionSchemeInfoBox` containing an `OMADRMKMSBox` SHOULD NOT be present in an `ItemProtectionBox`.

There is a difference between a streamable PDCF and a non-streamable PDCF. A streamable PDCF MUST conform to the server profile of the file format specification, and the media data is stored as packets. In a non-streamable PDCF, media data is stored as samples. An *access unit* is a group of one or more samples.

The encryption process changes both packet and sample formats from the original plaintext. The file format may support also other DRM key management systems than OMA DRM 2.2, but the encrypted access unit format is specific to OMA DRM. Thus, in the encryption process, non-streamable PDCFs MUST have the `OMADRMAUHeader` (see 7.1.6) inserted before each access unit.

7.1.3 DRM Scheme Type

The `SchemeTypeBox` includes information on which DRM system is being used to manage keys and decryption of the content. As the media file format MAY support also other key management systems than OMA DRM, the key management system in use is indicated by a 4CC in the *SchemeType* field [ISO14496-12].

Table 14: PDCF Scheme Type for OMA DRM

SchemeType	Value	Semantics
OMA DRM	'odkm'	OMA DRM is used for key management in the PDCF.

Table 15: PDCF Scheme Version for OMA DRM

SchemeVersion	Value	Semantics
OMA DRM 2.0	0x00000200	OMA DRM version is 2.0

For PDCF files conforming to this specification, the *SchemeType* MUST be the 4CC 'odkm', and *SchemeVersion* MUST be 0x00000200 (version 2.0). If OMA DRM key management scheme 'odkm' is indicated, then the file is a PDCF and MUST contain at least one `OMADRMKMSBox`. A PDCF MUST support only OMA DRM for the key management system.

7.1.4 Scheme Information

The `SchemeInformationBox ('schi')` is used to carry DRM key management system specific information, thus it is only a container box. For OMA DRM, this box MUST include exactly one `OMADRMKMSBox,` as the first sub-box.

7.1.5 OMA DRM Key Management System

There MAY be several instances of the `OMADRMKMSBox` in a PDCF file, exactly one per each protected track.

```
aligned(8) class OMADRMKMSBox extends FullBox('odkm', version, 0) {
      OMADRMCommonHeaders           Headers;              // Common headers box
      OMADRMAUFormatBox             AUFormat;
}
```

Table 16: OMA DRM Headers in PDCF

Field name	Type	Purpose
`Headers`	OMADRMCommonHeaders	OMA DRM Common headers as defined in 5.2.1.

Contained in the `OMADRMKMSBox` there MUST be:

- one `OMADRMCommonHeaders` box. The common headers box is exactly as defined in section 5.2.1
- one OMADRMAUFormatBox, as the second sub-box.

7.1.5.1 Common Headers

The OMADRM CommonHeaders box is the same as defined in section 5.2.1.However, in case of encryption using the AES_128_BYTE_CTR method, the ExtendedHeaders field in the OMADRMCommonHeaders box MUST include one instance of the OMADRMSalt box.

7.1.5.2 OMADRMSalt box

If the AES_128_BYTE_CTR encryption method is used, the ExtendedHeaders field in the OMADRMCommonHeaders box includes one instance of the OMADRMSalt box:

```
aligned (8) class OMADRMSalt extends Box('oslt') {
      unsigned int(8) SaltLength;          // Length of the Salt field in bits. MUST be 64
      unsigned int(SaltLength) Salt;       // Salt needed for AES_128_BYTE_CTR
}
```

The OMADRMSalt box contains the *Salt*, which is needed for AES_128_BYTE_CTR encryption method.

For the NULL, AES_128_CBC and AES_128_CTR encryption methods, the OMADRMSaltBox SHOULD NOT be included and has no meaning.

The Salt and IV MUST be used to encrypt the content as defined in [XBS-v1.0], section 13.4.1.2.

7.1.5.3 Access Unit Format Box

The OMADRMAUFormatBox is used to indicate the format of the `OMADRMAUHeader` which is placed on media access units.

```
aligned(8) class OMADRMAUFormatBox extends FullBox('odaf', 0, 0) {
      bit(1) SelectiveEncryption;
```

```
      bit(7) reserved;                    // Must be zero
      unsigned int(8) KeyIndicatorLength;  // Must be zero
      unsigned int(8) IVLength;
}
```

Where

SelectiveEncryption: Describes the use of Selective Encryption (refer [TS26.234] Annex K). If this field is set to 1, each OMADRMAUHeader contains the field EncryptedAU indicating whether the AU is encrypted or not. If the SelectiveEncryption field is set to 0, all AUs are encrypted and no EncryptedAU field appears in the OMADRMAUHeader.

IVLength: Describes the size of the initialisation vector (or initial counter value) in bytes. This length should be consistent with the algorithms used and indicated in Table 1.

KeyIndicatorLength: Describes the size of the key indicator in bytes. In this version of the specification, the value of KeyIndicatorLength is 0.

In case the `OMADRMAUFormatBox` is omitted the default values for the fields are:

SelectiveEncryption: 1 (enabled)

KeyIndicatorLength: 0

IVLength: default value is as per the encryption mode (see section 5.2.1.2)

7.1.6 Access Unit Format Header

The *Access Unit Format* specifies the format for each access unit protected by OMA DRM. A media file format specifies the layout of the media data as samples, but the encryption/decryption process requires additional information carried in each access unit. The additional information is dependent on the DRM key management used. OMA DRM specifies its own access unit header, which MUST precede the codec-specific sample data in each access unit.

```
aligned(8) class OMADRMAUHeader {
      if( SelectiveEncryption == 1 ) {
                  bit(1)  EncryptedAU;                // Encryption indicator
                  bit(7)  reserved;                   // Must be zero
      }
      else
             EncryptedAU = 1;
      if( EncryptedAU == 1 ) {
             unsigned int(8 * IVLength) IV;
             unsigned int(8 * KeyIndicatorLength) KeyIndicator;
      }
```

Note that the parameter “SelectiveEncryption” is described in the OMADRMAUFormatBox in Section 7.1.5.3.

Table 17: PDCF Access Unit Format

Field name	Type	Purpose
EncryptedAU	bit(1)	Encryption Indicator for the access unit.
IV	unsigned int(8 * IVLength)	Initialisation vector (or initial counter value)
KeyIndicator	unsigned int(8 * KeyIndicatorLength)	In this version of the

		specification, this field is not present as KeyIndicatorLength is zero.

Table 18: EncryptedAU Indicator values

EncryptedAU	Value	Semantics
None	0	Access unit is not encrypted.
Encrypted	1	Access unit is encrypted.

When encrypting PDCF Content, the `OMADRMAUHeader` information MUST be added to the processed access unit. Note that if the EncryptionMethod field in the OMADRMCommonHeaders box is set to NULL, both *SelectiveEncryption* and *IVLength* are set to 0. A playing Device uses the header information for decryption purposes and is able to extract the actual sample(s).

7.2 PDCF Streaming format (INFORMATIVE)

This section and its subsections are informative. This section describes how OMA DRM is applied to streaming content, especially in conjunction with a streaming service such as the 3GPP Packet switched Streaming Service (PSS) [TS26.234] or the 3GPP2 MSS [C.S0045].

Streaming DRM Content is leveraging the PDCF file format and widely deployed standard streaming protocols. DRM Content is transferred over a real time streaming protocol as encrypted packets, which include the original payload. The encrypted payload wrapper format MAY be used in any streaming service using RTSP streaming [RFC2326], SDP signalling [RFC2327] and RTP transport [RFC3550].

Supporting the PDCF streaming is OPTIONAL, even if PDCF format is supported. A multimedia streaming session MAY consist of protected PDCF tracks and unprotected tracks.

Streaming protected tracks is signalled through SDP parameters, using information contained in the sample format entries of the PDCF file. A streaming server derives network packets from a *hint track* in the media file.

The streamable PDCF profiles are defined by each service supporting OMA DRM. In conjunction with the streamable file format, an end-to-end streaming service such as [TS26.234] or [C.S0045] MUST specify the RTP payload format used and mechanisms for signalling OMA DRM and encryption parameters. This specification defines the OMA DRM parameters that MUST be signalled in a streaming session.

7.2.1 RTP Payload

The RTP payload format consists of two parts: the payload wrapper and the actual media payload. The media payload (e.g. H.263 video) is packetised according to the appropriate standard, encrypted as required, and stored as packets in the PDCF file. The encrypted payload wrapper includes a header with additional signalling information, such as *Selective Encryption* indicator and initial vector for the packet. With this mechanism, one encrypted payload specification is used to protect any standard RTP payload. Also a benefit of the wrapper format is that the DRM system is fully functional in networks supporting basic RTP profiles, and thus not placing requirements on existing network configurations.

7.2.2 Session signalling

For PDCF streaming, the session descriptors (SDP files) MUST include information about the wrapper payload. The format parameters for the wrapper format are used to signal e.g. DRM *Key Management Specific* parameters and *Encryption Parameters*.

Each streaming service supporting PDCF streaming must allocate space for signalling OMA DRM *Key Management Specific* headers. In the SDP *Encryption Parameters*, PDCF streaming MUST support the AES 128 cipher in counter mode. If the

Selective Encryption feature is disabled for a track, the Device MUST discard all packets belonging to this track where the encryption indicator is 'false' (unencrypted).

The *Key Management Specific* parameters MUST include the mandatory OMA DRM headers, as name value pairs. These parameters MUST be derived from the key management box in PDCF.

Table 19: Required OMA DRM specific parameters

Parameter name	Purpose
ContentID	ContentID for the protected track
RightsIssuerURL	The RightsIssuerURL for fetching Rights

Other headers MAY be added to the key management specific parameters, and a consuming Device MUST pass them to the DRM Agent. The DRM Agent will then act accordingly and acquire Rights for the stream as appropriate. The semantics of the headers are the same as the common headers defined in section 5.2.

8. MPEG-2 TS Profile (MPEG2DCF)

8.1 Introduction

ISO/IEC International Standard 13818-1 ([ISO/IEC 13818-1]), a.k.a MPEG2-System, specifies generic methods for multimedia multiplexing, synchronization and time base recovery. The Transport Stream (TS) specified therein is used as the stream format in many systems for digital television broadcast in the Consumer Electronics domain.

This specification defines a specific DRM Content Format profile that enables Content Providers to easily enhance their existing and deployed digital television services by allowing Devices with MPEG-2 TS support to access broadcast content. It enables Devices to request Rights Objects for access to broadcasted Media Objects that are transported to the home using an [ISO/IEC 13818-1] based content delivery system and then distributed to a Device using local connectivity. The figure below depicts a typical environment for delivery of broadcast content to the home using a Conditional Access (CA) terminal.

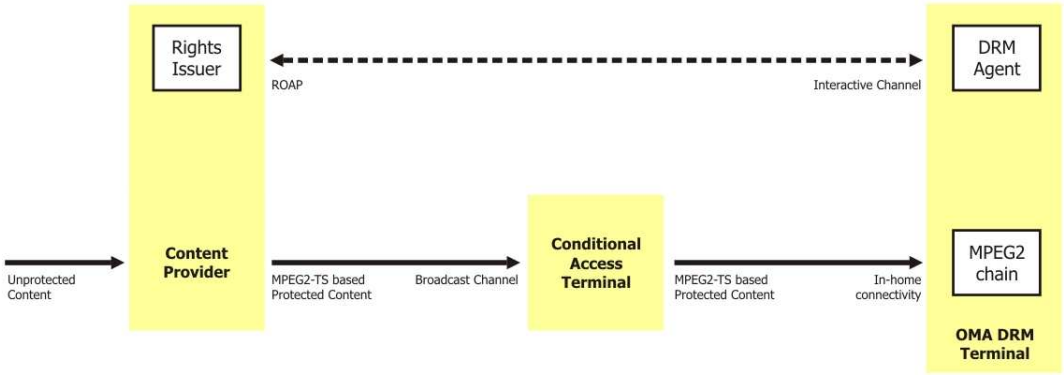


Figure 5: Typical environment for delivery of broadcast content to the home

This specification addresses the format of the broadcast content for access by a Device with MPEG-2 TS support, as well as protocols for communication between the Device and the Rights Issuer needed to access the broadcast content. The broadcast content delivery system and the Rights Issuer are beyond the scope of this specification. Hence this specification does neither require the presence nor the absence of a CA terminal. However, when a CA terminal is present, this specification allows that by means simulcrypt mechanisms beyond the scope of the specification the same broadcast content can be accessed in CA terminals (without OMA DRM support) and in Devices with MPEG-2 TS support conforming to this specification.

The specific DRM Content Format profile defined in this specification to support OMA DRM protected (broadcast) content contained in MPEG-2 transport streams is called the MPEG-2 transport stream DCF: MPEG2DCF. An MPEG2DCF is an MPEG-2 transport stream consisting of concatenated MPEG-2 transport packets that contain the (protected) broadcast content and the associated metadata needed to access the broadcast content. It should be noted that each MPEG-2 transport streams for broadcast is typically self-contained so as to allow for random access without out-of-band information. Thereby an MPEG2DCF can be considered as a file that typically carries all information needed to access the contained broadcast content.

When broadcast content is intended to be delivered to Devices, the Content Provider embeds the OMA specified information that is needed to create an MPEG2DCF file into the MPEG-2 TS prior to transport over the broadcast channel. The MPEG2DCF files may be broadcast directly to Devices or via CA terminals. When delivered via a CA-terminal, which is not necessarily OMA DRM conformant, then the CA terminal can create an MPEG2DCF file simply by storing the data received over the broadcast-channel into a file, without any modifications to the data. These MPEG2DCF files may then be distributed to Devices. To access the broadcast content contained in an MPEG2DCF, a Device needs to retrieve the associated Right Objects using OMA specified information in the MPEG2DCF.

To avoid the need for broadcasting content in multiple formats, the MPEG2DCF format re-uses commonly deployed methods for encryption of broadcast content. Various methods are defined by standardization bodies for broadcast applications, such as ARIB in Japan, ATSC in the USA and DVB in Europe. As a consequence, MPEG2DCF does not specify the encryption

method, but instead signals which encryption method is used. This has the important advantage that a Device may use commonly available hardware and software solutions for demultiplexing, decrypting and decoding of MPEG-2 transport

8.2 The MPEG-2 transport stream Profile of DCF (MPEG2DCF)

[ISO/IEC 13818-1] does not specify mechanisms for content or service protection. Instead it allows for (partial) encryption of the TS and it supports (simultaneous) implementation of content protection mechanisms through generic system data stream definitions. This infrastructure defined in [ISO/IEC 13818-1] is used by this specification as a basis for an OMA defined key management schema, potentially in addition to non-OMA CA systems used by CA terminals. This specification defines data structures for embedding into an MPEG2DCF as Entitlement Control Messages (see next sections). These data structures serve two purposes:

- To transport the information needed by a Device to request a Rights Object that is associated to the broadcast content, typically a program. This information corresponds to the information carried in the Common Headers defined in section 5.2.
- To transport the currently used content encryption key.

Although envisioned and enabled in [ISO/IEC 13818-1], this specification does not specify data structures to enable insertion of OMA DRM Rights Objects in the TS stream itself. Instead this specification envisions the Rights Objects to be retrieved via the interactive channel.

8.2.1 The MPEG-2 transport stream structure (Informative)

The Transport Stream defined in [ISO/IEC 13818-1] is a multiplex of packetized audio, video and other data, associated with (potentially) a number of simultaneously running programs and services. It consists of 188 bytes long Transport Stream packets, all of which have header with a Packet Identifier (PID). Typically the payload of all packets with the same PID constitutes a single elementary video-stream, audio-stream or data stream.

The data contained in TS packets with a PID of 0x00 constitutes the Program Association Table (PAT) - a directory of all currently embedded programs and services in the TS. Per program the PAT contains the PID value of the TS-packets that make up the Program Map Table (PMT) for that program. The PMT contains more information on a certain program, including (if applicable) content protection related information.

[ISO/IEC 13818-1] supports protection of content, for broadcast typically by means of Conditional Access (CA) systems, by allowing the encryption of the payload of TS-packets (see section 5.2). To increase the level of security, the encryption keys may be changed frequently. [ISO/IEC 13818-1] enables the signalling of TS-packet encryption and changes in encryption keys and it enables a Conditional Access system to embed “private” data into the TS to communicate key material to terminals.

For example, a broadcast may use “odd keys” and “even keys” to encrypt the content, with indication on transport packet level which key is used. Each key is used during a certain period. During the period that the "odd key" is used for decryption in the terminal, the next "even key" is conveyed in the transport stream, and when this period is elapsed, the transport packets indicate that the "even key" is to be used for decryption. Now the next "odd key" is conveyed in the transport stream to ensure its availability once the MPEG-2 TS packets indicate that the next "odd key" must be used. Etc. In this way the keys are provided in a timely manner, prior to the usage of the next key.

[ISO/IEC 13818-1] specifies two locations for CA systems to embed CA related information in the TS. Firstly there may be the Conditional Access Table (CAT), contained in TS-packets with a PID value of 0x01. This table contains, for each CA system that is enabled to provide access to the TS, a Conditional Access descriptors that contains the PID of the TS-packets that contain the Entitlement Management Messages (EMMs) for that CA system. The EMMs are intended to transport long-term key material similar to OMA DRM Rights Objects. Secondly there are Conditional Access descriptors located in the PMT of a program, containing the PID of the TS-packets that contain the Entitlement Control Messages (ECMs) for that program and a certain CA system. The ECMs are intended to transport short-term key material that allow for frequently changing content encryption keys.

There may be CA-descriptors for more than one Conditional Access system, all of which may then be able to provide access to the set of video, audio and data streams that constitute the program or service. This is usually referred to as “simulcrypt”.

If such simulcrypt is applied in a broadcast using key switching, then both CA systems must ensure that the keys are provided in a timely manner.

As Conditional Access systems are beyond the scope of MPEG-2 TS, [ISO/IEC 13818-1] only describes an overall structure for the EMMs and ECMs, leaving the specification of further details for encryption methods and key management to standardization bodies for broadcast systems and / or to proprietary systems.

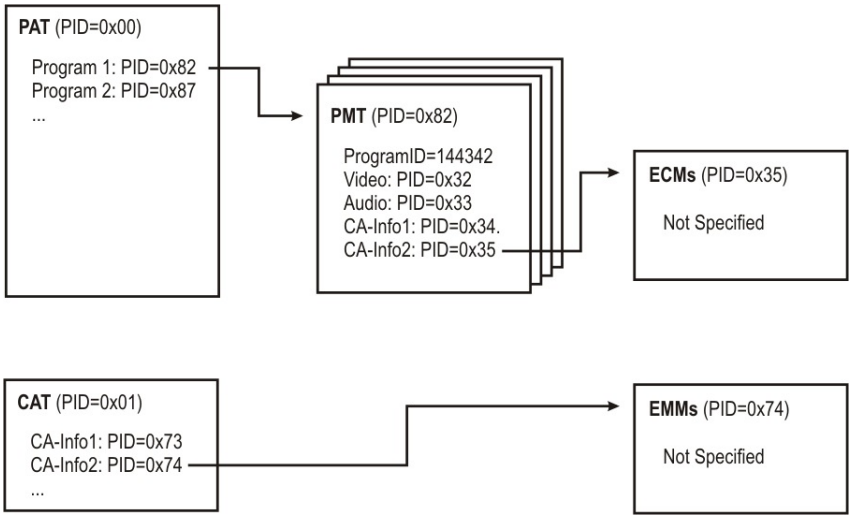


Figure 6: Example of the use Program Association Table (PAT), Program Map Table (PMT) and Conditional Access Table (CAT) to signal carriage of ECMs and EMMs

[ISO/IEC 13818-1] specifies which audiovisual streams are included within a program, but no further distinction is made about the content contained in a program. As a consequence, a program in an MPEG-2 TS may represent a broadcast concatenation of an episode of a comedy series, a sport event, a news show, a late night movie, etc. To identify the comedy episode, the sport event, the news show and the movie within an MPEG-2 TS defined program, requires tools not specified in [ISO/IEC 13818-1].

8.2.2 MPEG-2 transport stream scrambling

[ISO/IEC 13818-1] enables two ways to protect a stream: “Transport Stream level scrambling” and “PES level scrambling”.

MPEG-2 transport stream packets consist of a 4-byte header and a 184 byte data field. The data field can be subdivided between an adaptation field and a payload field. Depending on the size of the adaptation field, the length of the payload varies between 1 and 184 bytes. If TS level scrambling is used, only the payload of the transport stream packet is scrambled.

The MPEG2 PES packet consists of a PES header indicating among other things a variable length PES packet data block that follows the header bytes. The PES packet data block is carrying the payload. If PES level scrambling is used, only the payload is scrambled.

Independent of TS or PES level descrambling, the descrambler SHALL use content keys for descrambling scrambled (TS or PES) packets and the keys may be changed regularly.

8.2.2.1 Transport Stream level scrambling

To protect network broadcasts the broadcaster MAY "scramble" (a.k.a. encrypt) the transport stream (a.k.a. TS). In this case, scrambling takes place after multiplexing the payload of the transport packet. The receiving Device will have to "descramble" (a.k.a. decrypt) the TS so the audio and/or video and/or data parts can be consumed. This is typically done with a piece of hardware called the "descrambler". The descrambler is controlled by the Transport Stream Control (a.k.a. TSC) bits in the TS packet. These bits MUST be encoded as defined in **Table 20**. Limitations to TS level scrambling SHALL adhere to [ISO/IEC 13818-1].

Table 20– Definition of transport_scrambling_control bits

Transport Stream Control bits	Description
00	No descrambling (content not encrypted).
01	Reserved for use by broadcast applications.
10	Scrambling by the EVEN content key.
11	Scrambling by the ODD content key.

The encryption algorithms used SHALL be 128bit AES-CBC and 64bit DVB-CSA as described in section 8.2.4.1.

8.2.2.2 PES level scrambling

Instead of scrambling all the content at the TS level, one or more of the packetized elementary streams (a.k.a. PES) MAY be scrambled. In this case, scrambling generally takes place at the source, before multiplexing. The descrambler is controlled by the 2 bit PES scrambling control field in the PES packet header. These bits MUST be encoded as defined in **Table 21**. Limitations to PES level scrambling SHALL adhere to [ISO/IEC 13818-1].

Table 21– Definition of PES_scrambling_control field bits

PES_scrambling_control field	Description
00	No descrambling (content not encrypted).
01	No descrambling.
10	Scrambling by the EVEN content key
11	Scrambling by the ODD content key

8.2.2.3 Descrambling MPEG-2 content (Informative)

There are two possible descrambler implementations: single key and dual key descramblers.

- Single key descramblers have one register to store a descrambler key. To change the key they have to overwrite the key in the register. This cannot be done when the descrambler is in the middle of descrambling a packet. Therefore, the broadcaster usually alternates the scrambled transmission by transmitting packets in the clear a short interval (mostly 1 second). The descrambler’s register is then updated with the new key. After the clear interval expires, the broadcaster sends TS packet that are scrambled with the new key. This cycle can repeat itself endlessly.

- Dual key descramblers have two registers so they can store two keys: the first register can contain the key the descrambler is currently using and the second register can be updated with a new key for the next keying period. To distinguish the registers they are identified as the odd and even key register. The TSC bit in the TS packet indicates if the descrambler needs to use the key in the odd or even key register in order to descramble the TS packet and flips to corresponding register when necessary.

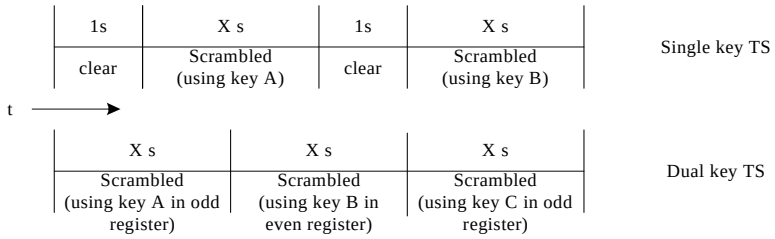


Figure 7 – Single key vs. dual key TS over time

The odd_even_flag in Key Stream Message ECM (see section 8.2.3.1) signals which content key is delivered. Because of the aforementioned behaviour, Table 22, applies.

Table 22 – Descrambling possibility matrix

•	• **Single key descrambler**	• **Dual key descrambler**
Single key TS	De facto	Possible [a]
Dual key TS	Impossible [b]	De facto

[a] TSC bits in single key TS will be stationary odd or even. Two solutions possible:
1) detect TSC status and put key in correct register, or
2) put the existing key in both the odd and even register at the same time.

[b] There is one second in the clear in dual key TS to change the key.

8.2.3 CA Descriptor usage in MPEG2DCF

This section defineds how to use the CA Descriptor, specified in [ISO/IEC 13818-1], within MPEG2DCF. Presence of a MPEG2DCF CA Descriptor signals transport of MPEG2DCF compliant ECMs and EMMs. The MPEG2DCF CA Descriptor SHALL be included in the PMT of each program, and MAY be included in the CAT. The DRM Agent SHALL support presence of MPEG2DCF CA Descriptor in the PMT and in the CAT

Table 23 defines the format of the CA descriptor for MPEG2DCF. The first fields in the MPEG2DCF CA Descriptor are specified in [ISO/IEC 13818-1], while the fields after the CA_PID field are MPEG2DCF specific.

Table 23 – MPEG2DCF CA Descriptor

Syntax	No. of bits	Mnemonic
MPEG2DCF_CA_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
CA_system_ID	16	uimsbf
MPEG2_reserved	3	bslbf

CA_PID
if (descriptor_length>4) {
for (i=0; I<N; i++){
CA_descriptor_message()
}
}
}
	13	uimsbf

descriptor_tag, MPEG has defined the value of 9 for the CA-descriptor.

descriptor_length, length of the descriptor.

CA_system_ID, the value of this field is 0x0008 assigned by DVB.

CA_PID, the PID on which the sections with MPEG2DCF ECMs/EMMs can be found

CA_descriptor_message(), see section 8.2.3.1.

8.2.3.1 CA_descriptor_message()

The CA descriptor message is a generic mechanism to extend MPEG2DCF CA descriptors. It consists of an identifier of the message, followed by the length of the message and message data bytes; see **Table 24**. An MPEG2DCF CA Descriptor MAY contain zero or more CA descriptor messages.

Table 24 – CA_descriptor_message

Syntax	No. of bits	Mnemonic
CA_descriptor_message() {		
CA_descriptor_message_ID	8	uimsbf
message_data_length	8	uimsbf
for (i=0; i< message_data_lengh); i++){		
message_data_byte	8	bslbf
}		
}		

CA_descriptor_message_ID, identifier of the type of the message as defined in **Table 25**.

Table 25 – CA_descriptor_message_ID values

Value	Description
0x00	Reserved
0x01	ServiceBaseCID
0x02	SocID
0x03 – 0x1F	Reserved for future use by OMA
0x20 – 0xFF	Available for private use by MPEG2DCF applications; values are assigned via OMNA registration

message_data_length, the message_data_length field specifies the number of message_data_ bytes immediately following this field.

message_data_byte, the coding and semantics of message_data_bytes depend on the value of the CA_descriptor_message_ID field.

8.2.3.1.1 ServiceBaseCID

ServiceBaseCID (Service Base Content ID) is the base part of the Content IDs of the programs and/or services contained in the MPEG2DCF. Content IDs MUST be globally unique URIs.

ServiceBaseCID MAY be included in the MPEG2DCF CA Descriptor or retrieved via unspecified out-of-band mechanism (e.g. in a service guide)

8.2.3.1.2 SocID

SocID bits represent the service operation center identifier for this content.

SocID MAY be included in the MPEG2DCF CA Descriptor or retrieved via unspecified out-of-band mechanism (e.g. in a service guide).

8.2.4 Usage of Private Sections in MPEG2DCF

The following sub-sections specify ECM and EMM messages for usage in MPEG2DCF. They are formatted as so-called private sections defined in [ISO/IEC 13818-1]. A stream of such private sections may be used to transport a number of logical datastructures, called tables, with a certain frequency. Each ECM and EMM contributes to a certain table, as identified by the table identifier (table_id) field in the ECM and EMM. Allocation of table identifiers for use in MPEG2DCF is done in **Table 26**.

Table 26 – Allocation of ECM and EMM table identifiers in MPEG2DCF

Table_id	Table description
0x80	Key Stream Message ECM
0x81	Key Stream Message ECM
0x82	Rights URL EMM
0x83	Textual headers EMM
0x84	Extended headers EMM
0x85	Enforced Advertising Service ECM

8.2.4.1 Key Stream Message EMM

Key Stream Message (KSM) defined in section 7.2 of [IEC62455] SHALL be used to carry MPEG2DCF encryption key material. KSM MUST be carried in MPEG2DCF KSM section defined in **Table 27.**

Table 27 – MPEG2DCF KSM section

Syntax	No. of bits	Mnemonic	Value
MPEG2DCF_KSM_section() {			
table_id	8	uimsbf	0x80 or 0x81
section_syntax_indicator	1	bslbf	0
OMA_reserved	1	bslbf	
MPEG2_reserved	2	bslbf	
section_length	12	uimsbf	
for (i=0; i<N; i++) {			
ksm_data_byte	8	uimsbf	
}			
}			

table_id, identifies the section as MPEG2DCF KSM.

section_syntax_indicator, set to 0 to signal the use of the short section header ([ISO/IEC 13818-1], section 2.4.4.11).

OMA_reserved, bit reserved for future use by OMA.

MPEG2_reserved, bits reserved by [ISO/IEC 13818-1].

section_length, specifies number of bytes that follow the section_length field up to the end of the section.

ksm_data_byte, KSM defined in section 7.2 of [IEC62455] with the restrictions defined below.

The following restrictions SHALL apply to IEC 622455 KSM:

- traffic protection protocol is set to KSM_ALGO_MPEG2_TS_CRYPT
- traffic_authentication_flag is set to KSM_FLAG_FALSE
- two kinds of encryption algorithms are supported in MPEG2DCF: 128bits AES and 64bit DVB-CSA
- KSM_ALGO_MPEG2_TS_CRYPT specific fields of IEC 62455 KSM MUST take the following values
  - In case of 128bit AES encryption

    content_key_index, set to '0x5'.

    cipher_mode, set to 0x1 (CBC mode).

    initial_vector_length, set to 16 (128bits).

    initial_vector, carries initial encryption vector.
  - In case of 64bit DVB-CSA encryption

    content_key_index, set to '0x0'.

    cipher_mode, set to 0x2 (CSA mode).

- encrypted_traffic_key_material_length is set to 16 (128bits)
- permissions_flag is set to 0
- access_criteria_descriptor loop MAY contain access criteria descriptors. The DRM Agent SHALL ignoreaccess criteria descriptors that it does not support unless specified otherwise. If the semantics of an available access criteria descriptor overlap with one or more permissions or constraints defined in [DRMREL-v2.2], this access criteria descriptor SHALL be ignored by the DRM Agent.

8.2.4.1.1 Content_control_information access criteria descriptor

This section defines an access_criteria_descriptor that MAY be present in the IEC 62455 ECM as defined in section 8.2.4.1.

The content_control_information access criteria descriptor is used for signaling access requirements to be fulfilled before consumption of the associated multicast content can be continued. In DRM v2.2, it is used to carry information related to mandatory play out of advertisements (see [DRM-v2.2]). For instance, it allows indicating that that the chunk of the MPEG2 transport stream following Key Stream Message ECM containing content_control_information access criteria descriptor is an advertisement that must not be skipped or fast forwarded.

Table 28– content_control_information access criteria descriptor

content_control_information access_criteria_descriptor	Tag	Length	Type
content_control_information_byte	0x11	8	bslbf

content_control_information_byte, is defined in **Table 29**.

Note: The DRM agent that does not support content_control_information access criteria descriptor MUST not allow rendering the associated stream.

Table 29 – Syntax of content_control_information_byte

Syntax	No. of bits	Mnemonic
content_control_information_byte() {		
lock_controls_bit	1	bslbf
reserved_bits	7	bslbf
}		

lock_controls_bit, if the value of this bit is set to "1" the content received after Key Stream Message ECM MUST not be skipped or fast forwarded (i.e. user controls of the content MUST be locked by the DRM Agent or media player); if the value of this bit is set to "0", the user controls of the content (i.e. skip or fast forward) MUST be unlocked.

reserved_bits, bits reserved for use in the future versions of this specification.

8.2.4.2 Rights URL EMM

The Rights URL EMM contains information potentially needed by a DRM Agent to request a Rights Object for the MPEG2DCF. MPEG2DCF MAY carry the Rights URL EMM. The DRM Agent supporting MPEG2DCF SHALL support receipt of the Rights URL EMM..

Table 30 – MPEG2DCF Rights URL section

Syntax	No. of bits	Mnemonic	Value
MPEG2DCF_Rights_URL_section() {			
table_id	8	uimsbf	0x82
section_syntax_indicator	1	bslbf	0
OMA_reserved	1	bslbf	
MPEG2_reserved	2	bslbf	
section_length	12	uimsbf	
for (i=0; i<N; i++){			
Rights_URL_message()			
}			
}			

table_id, identifies the section as RI URL EMM. See **Table 26**.

section_syntax_indicator, set to 0 to signal the use of the short section header ([ISO/IEC 13818-1] section 2.4.4.11).

OMA_reserved, bit reserved for future use by OMA.

MPEG2_reserved, bits reserved by [ISO/IEC 13818-1].

section_length, the number of bytes that follow the section_length field up to the end of the section.

Rights_URL_message(), one or more messages containing URLs required to request a Rights Object. The general structure of RI_URL_message is provided in section 8.2.4.2.1.

8.2.4.2.1 Rights_URL_message()

The Rights_URL_message() is a mechanism to include URLs in a Rights URL EMM. Each message consists of a message identifier, followed by the length of the message and message data bytes; see **Table 31**.

Table 31 – Rights_URL_message

Syntax	No. of bits	Mnemonic
Rights_URL_message() {		
rights_URL_message_ID	8	uimsbf
message_data_length	8	uimsbf
for (i=0; i< message_data_lengh); i++){		
message_data_byte	8	bslbf
}		
}		

rights_URL_message_ID, identifies type of the contained URL as specified in**Table 32**.

Table 32 – Rights_URL_message_ID values

Value	Description
0x00	RightsIssuerURL(see section 8.2.4.2.1.1)
0x01	SilentRightsURL(see section 8.2.4.2.1.2)
0x02	PreviewRightsURL(see section 8.2.4.2.1.3)
0x03 – 0x1F	Reserved for future use by OMA
0x20 – 0xFF	Available for private use by MPEG2DCF applications; values are assigned via OMNA registration

message_data_length, the message_data_length field specifies the number of message_data_ bytes immediately following this field.

message_data_byte, the coding and semantics of message_data_bytes depend on the value of the rights_URL_message_ID field (see **Table 32**).

8.2.4.2.1.1. RightsIssuerURL

Rights Issuer URL for this content as specified in section 5.2.1.9. Rights_URL_message() MAY carry the RightsIssuerURL.

8.2.4.2.1.2. SilentRightsURL

Silent Rights URL for this content. Rights_URL_message() MAY carry the SilentRightsURL.

SilentRightsURL SHALL be encoded on N+1 bytes as follows:

silent-method | silent-rights-url

silent-method (1 byte), as defined in section 5.2.2.1, where “on-demand” SHALL be encoded as 0x00 and “in-advance” SHALL be encoded as 0x01.

silent-rights-url (N bytes), as defined in section 5.2.2.1.

8.2.4.2.1.3. PreviewRightsURL

The Preview Rights URL for this content as specified in section 5.2.2.2. Rights_URL_message() MAY carry PreviewRightsURL.

8.2.4.3 Textual Headers ECM

The Textual Headers EMM contains additional information about the content. MPEG2DCF MAY carry the Textual Headers EMM. The DRM Agent supporting MPEG2DCF MAY support receipt of the Textual Headers EMM.

Table 33 – MPEG2DCF Textual header section

Syntax	No. of bits	Mnemonic	Value
MPEG2DCF_textual_header_section() {			
table_id	8	uimsbf	0x83
section_syntax_indicator	1	bslbf	1
OMA_reserved	1	bslbf	
MPEG2_reserved	2	bslbf	
private_section_length	12	uimsbf	
table_id_extension	16	uimsbf	
MPEG2_reserved	2	bslbf	
version_number	5	uimsbf	
current_next_indicator	1	bslbf	
section_number	8	uimsbf	
last_section_number	8	uimsbf	
for (i = 0; i < private_section_length-9;.i) {			
textual_header_byte	8	bslbf	
}			
CRC_32	32	rpchof	
}			
}			

table_id, identifies the section as Textual Header EMM. See **Table 26**.

section_syntax_indicator, set to 1 to signal the use of the long section header ([ISO/IEC 13818-1] section 2.4.4.11).

OMA_reserved, bit reserved for future use by OMA.

MPEG2_reserved, bits reserved by [ISO/IEC 13818-1].

section_length, the number of bytes that follow the section_length field up to the end of the section.

version_number, the version_number of the table ([ISO/IEC 13818-1] section 2.4.4.11).

section_number, the section_number of the table ([ISO/IEC 13818-1] section 2.4.4.11).

last_section_number, the last_section_number of the table ([ISO/IEC 13818-1] section 2.4.4.11).

textual_header_byte, the textual header information. Each textual header section will carry one textual header as specified in section 5.2.2.

8.2.4.4 Extended Headers EMM

The Extended Headers EMM contains extended headers defined in section 5.2.3. MPEG2DCF MAY carry the Extended Headers EMM. The DRM Agent supporting MPEG2DCF MAY support receipt of the Extended Headers EMM.

Table 34 – MPEG2DCF Extended header section

Syntax	No. of bits	Mnemonic	Value
MPEG2DCF_extended_header_section() {			
table_id	8	uimsbf	0x84
section_syntax_indicator	1	bslbf	1
OMA_reserved	1	bslbf	
MPEG2_reserved	2	bslbf	
private_section_length	12	uimsbf	
table_id_extension	16	uimsbf	
MPEG2_reserved	2	bslbf	
version_number	5	uimsbf	
current_next_indicator	1	bslbf	
section_number	8	uimsbf	
last_section_number	8	uimsbf	
for (i = 0; i < private_section_length-9;,i) {			
extended_header_byte	8	bslbf	
}			
CRC_32	32	rpchof	
}			
}			

table_id, identifies the section as Extended Headers EMM. See **Table 26**.

section_syntax_indicator, set to 1 to signal the use of the long section header ([ISO/IEC 13818-1] section 2.4.4.11).

OMA_reserved, bit reserved for future use by OMA.

MPEG2_reserved, bits reserved by [ISO/IEC 13818-1].

section_length, the number of bytes that follow the section_length field up to the end of the section.

version_number, the version_number of the table ([ISO/IEC 13818-1] section 2.4.4.11).

section_number, the section_number of the table ([ISO/IEC 13818-1] section 2.4.4.11).

last_section_number, the last_section_number of the table ([ISO/IEC 13818-1] section 2.4.4.11).

extended_header_byte, the extended header information. Each extended header section will carry one extended header as specified in section 5.2.3.

8.2.4.5 Enforced Advertising Service ECM

The Enforced Advertising Service ECM contains part of the information needed by a DRM Agent to execute the Enforced Advertising for the MPEG2DCF. See **Table 35**.

Table 35 – Enforced Advertising Service section

Syntax	No. of bits	Mnemonic	Value
Enforced_Advertising_Service_section() {			
table_id	8	uimsbf	0x85
section_length	12	uimsbf	
last_continuity_counter	4	uimsbf	
reserved	8	uimsbf	
for (i=0; i<N; i++){			
Enforced_Advertising_policy()			
}			
}			

table_id, identifies the section as Enforced Advertising Service ECM. See **Table 26**.

section_length, the number of bytes that follow the section_length field up to the end of the section.

last_continuity_counter, the continuity_counter value of the last transport packet before the rendering of the Enforced Advertising, indicating the start time of the Enforced Advertising.

Enforced_Advertising_policy (), the rules for Enforced Advertising, the general structure is provided as following:

```
    Enforced_Advertising_Policy() {
       NumberofAd             8         bslbf
       for ( i = 0 ; i < NumberofAd ; i++ ) {
            Advertisement_Content() //asset information
            EnforcedAdvertising()
       }
    }

    Advertisement_Content(){
       length            16             bslbf
       for ( i = 0 ; i < length ; i++ ) {
            byte  8        uimsbf        // Asset id
       }
    }
```

Note:
1)the Advertisement_Content()may contain the URL for advertising source from where the terminal will get the advertisement content if the advertisement content is not contained in the MPEGTSDCF.
2)the enforced advertisement content may be in another stream(not the same stream as current program stream) and the Advertisement_Content()will be used to indicate the stream id of the advertisement stream.

```
    EnforcedAdvertising(){
       length            16             bslbf
       playoutPresent     1             bslbf
       displayoutPresent  1             bslbf
       executeoutPresent  1             bslbf
       reserved           5             bslbf

       if ( playoutPresent ) {
             Policy()
       }
       if ( displayoutPresent ) {
             Policy()
       }
       If(executeoutPresent){
           Policy()
       }

    }

    Policy(){
       enforcement-count     8     uimsbf
       enforcement-duration  16    uimsbf     // the time duration for
                          //the Enforced Advertising with the unit of second.
       gracetimePresent   1             bslbf
       reserved           7             bslbf
       If (gracetimePresent)
       grace-time                       16    uimsbf // the minimum period of time
                                 //(in seconds) before enforcement-duration ends
    }
```

8.2.5 Accessing the MPEG2DCF

To access the MPEG2DCF, the Device must perform the following steps:

1. Parse the MPEG-2 transport stream to identify whether it is MPEG2DCF compliant by means of the MPEG2DCF CA descriptor.
2. If the MPEG-2 transport stream is MPEG2DCF compliant, then parse the MPEG2DCF to retrieve the MPEG2DCF related EMMs and/or ECMs.
3. If the Rights Object for accessing the MPEG2DCF is not available, then request this Rights Objects from the Rights Issuer.
4. Process the Rights Objects and ECM's to descramble the MPEG-2 transport stream.

The first two steps in this process involve retrieving and processing the PAT, PMT, CAT, MPEG2DCF CA Descriptor and retrieving the MPEG2DCF related ECMs. This process is specified in [ISO/IEC 13818-1] (also see section 8.2.1). The other steps are described in this section.

8.2.5.1 Requesting the Rights Object

The DRM Agent can use Rights Issuer URL or Silent Rights URL contained in Rights_URL_message() in order to initiate acquisition of the Rights Object. If both URLs are included in the message, the parameter appearing first MUST be prioritized and used to attempt to acquire the RO.

In order to create a request for RO, it is required to construct Content ID of the content (program or service) first.

The identifier of a program contained in the MPEG2DCF SHALL be constructed as follows:

CID="cid:"||socID||"#P"||serviceBaseCID||"@"||hex(program_CID_extension)

The identifier of a service contained in the MPEG2DCF SHALL be constructed as follows:

CID="cid:"||socID||"#S"||serviceBaseCID||"@"||hex(service_CID_extension)

The socID and serviceBaseCID values are retrieved from the MPEG2DCF CA descriptor or via out-of-band method. The program_CID_extension and service_CID_extension are retrieved from Key Stream Message ECM defined in section 8.2.4.1.

The Content IDs used for the programs and service contained in the MPEG2DCF MUST be globally unique URIs and MUST comply to the requirements for Contents IDs defined by this specification.

Note that, alternatively, the Device may use Rights Issuer URL to initiate browsing session with the Rights Issuer to purchase a Rights Object as defined in [DRM-v2.2].

8.2.5.2 Processing the Rights Object and KSM

From the acquired RO, the DRM Agent MUST retrieve CEK and if available the MAC key. If present in the RO, the MAC key MUST be used to authenticate the Key Stream Message ECM via program_MAC or service_MAC fields as defined in [IEC62455]. If MAC validation ends with failure, the DRM Agent SHALL disregard the ECM.

The CEK MUST be used as Program Encryption Key or Service Encryption Key defined in [IEC62455] when KSM ECM is associated to the program or service respectively.

8.2.6 Exchanging an MPEG2DCF between Devices

An MPEG2DCF may be exchanged between Devices. As described in previous sections, an MPEG2DCF must be parsed to determine its contents and a Rights Object must be requested to access the MPEG2DCF. The Rights Object that is received may be a domain Rights Object, which may be usable on other Devices. This section specifies means that enable efficient exchange between Devices of the information extracted from the MPEG2DCF and the Rights Objects retrieved from an RI for the MPEG2DCF. This is achieved by creating (P)DCF files for the MPEG2DCF file. In this way not all Devices that want to gain access the MPEG2DCF are required to perform all of the processing described in previous sections.

Before creating (P)DCF's for an MPEG2DCF, a Device MAY split-up a single large MPEG2DCF into a number of separate MPEG2DCF files (e.g.to locate each program in the MPEG2DCF into a separate file).

A Device MAY associate a (P)DCF with an MPEG2DCF in one of two ways. When using a DCF, the MPEG2DCF is either embedded into the DCF as a Content Object, or the MPEG2DCF file is referenced from the DCF using a Content Location Header. A PDCF may be used in applications that require synchronization of MPEG2DCF contents with other data (e.g. timed meta-data that is received in parallel to the MPEG2DCF or generated during recording). In this case the MPEG2DCF is an ISO File Format hint track in the PDCF. The standard mechanisms of the ISO File Format allow an embedded track in the PDCF or referencing the MPEG2DCF file.

When using a DCF, it is RECOMMENDED that a Device creates a separate DCF-file for each program in the MPEG2DCF, irrespective whether the part of the MPEG2DCF that contains that program is embedded into the DCF-file itself as a Content Object or stored as a separate file that is referenced from the DCF. An Device MAY also create a multi-part DCF for an MPEG2DCF, such that each part of the multi-part DCF is associated with one program in the MPEG2DCF.

It is also RECOMMENDED that all valid Rights Objects that were received for that program of the MPEG2DCF are embedded into the Mutable DRM Info Box of the (P)DCF-file.

8.2.6.1 ContentID field in the DCF header

The ContentID field can either be filled with a Service_CID or a Program_CID, as specified in previous sections. If available, the ContentID field SHOULD be filled with a Program_CID.

8.2.6.2 Content-Location header in the DCF header

The Content-Location, indicating which part of the MPEG2DCF to access for referenced content, is specified as follows.

```
ContentLocation = “Content-Location” “:” content-uri
content-uri = token | ( token || “#” || start_byte || end )
end = (“-” || end_byte) | ( “+” || n_bytes)
start_byte = *digit
end_byte = *digit | “end”
n_bytes = *digit
```

Where

token MUST be a file name, relative to the location of the DCF, or, if token is the empty string, the Content-Location header refers to the MPEG2DCF that is contained in the DCF file itself,

start_byte is the index of the byte in the MPEG2DCF to access the referenced content (program or service); the index of the first byte of the MPEG2DCF has value 0,

end_byte is either the string “end”, indicating the last byte of the file, or a number consisting of one or more digits, indicating the index of a byte in the MPEG2DCF after which no more bytes of the referenced content (program or service) are contained in the MPEG2DCF

n_bytes is the number of bytes in the MPEG2DCF immediately following start_byte, after which no more bytes of the referenced content (program or service) are contained in the MPEG2DCF.

8.2.6.3 EncryptionMethod field in the DCF header

In this specification, we specify a new value for Algorithm-id for use in the EncryptionMethod field in a DCF header.

Algorithm-id: MPEG-2_transport_stream_encryption

Value: for MPEG-2_transport_stream_encryption, the value 0x04 is assigned by OMNA

Semantics: the content is an MPEG Transport Stream which is protected as specified in Section 8.2.4.1

8.2.7 Enforced Advertising

To conduct the Enforced Advertising, the Device must perform the following steps:

1. Parse the MPEG2DCF to retrieve the Enforced Advertising Service ECM.
2. Parse the Enforced Advertising Service ECM to conduct the Enforced Advertising.

Once parsing the last PES packet with the program_packet_sequence_counter value equal to last_PES packet_sequence_counter, the Enforced Advertising is performed after. During the Enforced Advertising, the rendering is forbidden of fast forward or back forward.

If the advertisement content is not in the same stream of current program content, during the Enforced Advertising, the program stream may just contain non-meaning package.

The enforced advertisement content may be multimedia content or a static graphic.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-TS-DRM-DCF-V2_1-20081014-A	14 Oct 2008	Status changed to approved by TP ref# OMA-TP-2008-0382-INP_DRM_V2_1_ERP_for_Final_Approval

A.2 Draft/Candidate Version History

Document Identifier	Date	Description
Draft Versions OMA-TS-DRM-DCF-V2_2	27 Mar 2010	Initial draft of DRM DCF 2.2
	07 July 2010	Agreed CR in the CC in 24 June, OMA-DRM-2010-0103R02-CR_Incorporate_MDCF_into_DCF
	07 July 2010	Agreed CR in the CC in 8 Aprail, OMA-DRM-2010-0098R01-CR_AES128_BYTE_CTR_encryption_method
	06 Aug 2010	Agreed CRs in the CCs in 22 July and 5 August, OMA-DRM-2010-0173R01-CR_DCF_2.2_Introduction_Complement ; OMA-DRM-2010-0174R01- CR_DCF2.2_Description_Update_of_DRM_Content_Format ; OMA-DRM-2010-0176-CR_Replace_MDCF_with_MPEG2DCF
	03 Sep 2010	Agreed CRs in the Budapest meeting in September, OMA-DRM-2010-0182R01-CR_DCF_Introducing_IEC_62455_ECM OMA-DRM-2010-0183-CR_DCF_Clerical_Changes_August2010
	09 Oct 2010	Agreed CRs in the CCs from September till Oct 7th, OMA-DRM-2010-0193R01-CR_DCF_advertisements_control_ECM OMA-DRM-2010-0198-CR_Enforced_Advertising_for_MPEGTSDCF OMA-DRM-2010-0199R01-CR_DCF2.2_Format_of_advertisement
	22 Oct 2010	Agreed CRs in the CCs in Oct 14th and 21st:, OMA-DRM-2010-0186-CR_DCF_extension_for_dynamic_Advertisements_update OMA-DRM-2010-0204-CR_DCF_key_stream_message_ECM_update OMA-DRM-2010-0210- CR_DCF2.2_supplement_for_EnforcedAdvertising_elements OMA-DRM-2010-0211R02-CR_DCF2.2_supplement_for_SCR
	25 Feb 2011	Agreed CRs in the Hawaii meeting and CCs: OMA-DRM-2011-0007-CR_DCF_Fix_content_identification_in_MPEG2DCF OMA-DRM-2011-0008-CR_DCF_Convert_ECMs_to_EMMs_in_MPEG2DCF OMA-DRM-2011-0009-CR_DCF_Requesting_RO_in_MPEG2DCF OMA-DRM-2011-0015-CR_DCF_Transport_Stream_scrambling_in_MPEG2DCF OMA-DRM-2011-0016-CR_DCF_minor_editorials OMA-DRM-2011-0027R01- CR_DCF2.2_Dynamic_Advertisement_Update_AP_1145 OMA-DRM-2011-0044-CR_DCF2.2_To_Close_AP1158
	09 Mar 2011	Updates in terms of the decisions in CONRR.
	10 Mar 2011	According to 2011-CR-0047
	06 Apr 2011	Replaced TBD with CA_System_ID value “0x0008” assigned by DVB on April 6, 2011 (see http://www.dvbservices.com/identifiers/ca_system_id).
Candidate Version OMA-TS-DRM-DCF-V2_2	19 Apr 2011	Status changed to Candidate by TP TP ref # OMA-TP-2011-0136-INP_DRM_V2_2_ERP_for_Candidate_Approval

Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [IOPPROC].

B.1 Client Conformance Requirements

Item	Function	Reference	Status	Requirement
DRM--CLI-001	DCF support	6	M	
DRM--CLI-002	PDCF support	7	O	(DRM--CLI-103 AND DRM--CLI-104 AND DRM--CLI-105 AND DRM--CLI-106 AND DRM--CLI-107 AND DRM--CLI-108 AND DRM--CLI-109 AND DRM--CLI-110 AND DRM--CLI-111 AND DRM--CLI-112 AND DRM--CLI-114) AND (DRM--CLI-101 OR DRM--CLI-102)
DRM-CLI-003	MPEG2DCF support	8	O	DRM--CLI-201 and DRM--CLI-202 and DRM--CLI-203 and DRM--CLI-204 and DRM--CLI-205 and DRM--CLI-206 and DRM--CLI-207
DRM--CLI-004	AES_128_CBC encryption algorithm	5.2.1.2	M	
DRM--CLI-005	AES_128_CTR mode encryption algorithm	5.2.1.2	O	
DRM--CLI-006	Ignore unsupported boxes	5.1	M	
DRM--CLI-007	Common headers	5.2.1	M	
DRM--CLI-008	Textual headers	5.2.2	M	
DRM--CLI-009	GroupID	5.2.3.1	M	
DRM--CLI-010	Mutable DRM Information box	0	M	
DRM--CLI-011	Transaction Tracking box	5.2.4.1	M	
DRM--CLI-012	Rights Object box	5.2.4.2	M	
DRM--CLI-013	ISO format constraints	6.2.1	M	
DRM--CLI-014	FullBox version	6.2.1	M	
DRM--CLI-015	DCF header	6.2.2	M	
DRM--CLI-016	OMA DRM container box	6.3.1	M	
DRM--CLI-017	Discrete headers box	6.3.2	M	
DRM--CLI-018	User-Data box in Discrete headers box	6.3.2.3	M	
DRM--CLI-019	Content Object box	6.3.2.3.4	M	
DRM--CLI-020	Multipart DCF	6.4	M	
DRM--CLI-021	Extension boxes	0	O	
DRM--CLI-022	UTF-8 character encoding for 3GPP asset information	6.3.2.3	M	
DRM--CLI-023	UTF-16 character encoding for 3GPP asset information	6.3.2.3	O	
DRM--CLI-024	User-Data box in Mutable DRM Information box	5.2.4.3	O	
DRM--CLI-025	AES_128_BYTE_CTR mode encryption algorithm	5.2.1.2	O	

B.2 Client Conformance Requirements For The PDCF Format

Item	Function	Reference	Status	Requirement

Item	Function	Reference	Status	Requirement
DRM--CLI-101	3GPP conformance	7	O	Conform to [TS26.244]
DRM--CLI-102	3GPP2 conformance	7	O	Conform to [C.S0050]
DRM--CLI-103	OMA DRM key management	7.1.5	O	
DRM--CLI-104	OMA DRM scheme	7.1.3	O	
DRM--CLI-105	Common headers	5.2.1	O	
DRM--CLI-106	AES_128_CTR mode encryption algorithm	5.2.1.2	O	
DRM--CLI-107	Textual headers	5.2.2	O	
DRM--CLI-108	GroupID	5.2.3.1	O	
DRM--CLI-109	Mutable DRM Information box	0	O	
DRM--CLI-110	Transaction Tracking box	5.2.4.1	O	
DRM--CLI-111	Rights Object box	5.2.4.2	O	
DRM--CLI-112	OMA DRM access unit format	7.1.6	O	
DRM--CLI-113	User-Data box in Mutable DRM Information box	5.2.4.3	O	
DRM--CLI-114	AES_128_BYTE_CTR mode encryption algorithm	5.2.1.2	O	

B.3 Client Conformance Requirements For The MPEG2DCF Format

Item	Function	Reference	Status	Requirement
DRM--CLI-201	Descrambling of TS	8.2.2	O	
DRM--CLI-202	CA descriptor usage in MDCF	8.2.3	O	
DRM--CLI-203	Usage of private sections in MDCF	8.2.4	O	
DRM--CLI-204	Key Stream Message ECM	8.2.4.1	O	
DRM--CLI-205	Enforced Advertising	8.2.4.6;8.2.7	O	
DRM--CLI-206	Accessing the MDCF	8.2.5	O	
DRM--CLI-207	Exchanging MDCF between Devices	8.2.6	O	

Appendix C. Reserved Numbers (Informative)

This Appendix lists common 4CC constants used in DCF and PDCF formats. The tables list only 4CC constants specified by OMA.

Table 36: Reserved identifier constants in the DCF format

4CC	Reference	Purpose
‘ohdr’	5.2.1	Common headers box
‘mdri’	0	Mutable DRM Information box
‘grpi’	5.2.3.1	Group ID box
‘odtt’	5.2.4.1	Transaction Tracking box
‘odrb’	5.2.4.2	Rights Object box
‘odcf’	6.2.2	File brand
‘odrm’	6.3.1	OMA DRM Container box
‘odhe’	6.3.2	Headers box for the Discrete Media profile box
‘icnu’	6.3.2.3.2	Icon URI
‘infu’	6.3.2.3.3	Info URL
‘cvru'	6.3.2.3.4	Cover URI
‘lrcu'	6.3.2.3.5	Lyrics URI
‘odda’	6.3.2.3.4	Content Object box

Table 37: Reserved OMA DRM specific identifier constants in the PDCF format

4CC	Reference	Purpose
‘grpi’	5.2.3.1	Group ID box
‘mdri’	0	Mutable DRM Information box
‘odtt’	5.2.4.1	Transaction Tracking box
‘odrb’	5.2.4.2	Rights Object box
‘ocid’	5.2.4.3.1	ContentID box
‘icnu’	6.3.2.3.2	Icon URI
‘infu’	6.3.2.3.3	Info URL
‘ocru'	6.3.2.3.4	Cover URI
‘olcu'	6.3.2.3.5	Lyrics URI
‘odkm’	7.1.4, 7.1.5	OMA DRM scheme type, OMA DRM scheme information box identifier

‘ohdr’	7.1.5.1	Common headers box
‘odaf’	7.1.5.3	Access Unit Format box