



DRM Content Format

Draft Version 2.0 – 03-November-2003

Open Mobile Alliance
OMA-DRM-DCF-v2_0-20031103-D

Use of this document is subject to all of the terms and conditions of the Use Agreement located at <http://www.openmobilealliance.org/UseAgreement.html>.

Unless this document is clearly designated as an approved specification, this document is a work in process, is not an approved Open Mobile Alliance™ specification, and is subject to revision or removal without notice.

You may use this document or any part of the document for internal or educational purposes only, provided you do not modify, edit or take out of context the information in this document in any manner. Information contained in this document may be used, at your sole risk, for any purposes. You may not use this document in any other manner without the prior written permission of the Open Mobile Alliance. The Open Mobile Alliance authorizes you to copy this document, provided that you retain all copyright and other proprietary notices contained in the original materials on any copies of the materials and that you comply strictly with these terms. This copyright permission does not constitute an endorsement of the products or services. The Open Mobile Alliance assumes no responsibility for errors or omissions in this document.

Each Open Mobile Alliance member has agreed to use reasonable endeavors to inform the Open Mobile Alliance in a timely manner of Essential IPR as it becomes aware that the Essential IPR is related to the prepared or published specification. However, the members do not have an obligation to conduct IPR searches. The declared Essential IPR is publicly available to members and non-members of the Open Mobile Alliance and may be found on the “OMA IPR Declarations” list at <http://www.openmobilealliance.org/ipr.html>. The Open Mobile Alliance has not conducted an independent IPR review of this document and the information contained herein, and makes no representations or warranties regarding third party IPR, including without limitation patents, copyrights or trade secret rights. This document may contain inventions for which you must obtain licenses from third parties before making, using or selling the inventions. Defined terms above are set forth in the schedule to the Open Mobile Alliance Application Form.

NO REPRESENTATIONS OR WARRANTIES (WHETHER EXPRESS OR IMPLIED) ARE MADE BY THE OPEN MOBILE ALLIANCE OR ANY OPEN MOBILE ALLIANCE MEMBER OR ITS AFFILIATES REGARDING ANY OF THE IPR'S REPRESENTED ON THE “OMA IPR DECLARATIONS” LIST, INCLUDING, BUT NOT LIMITED TO THE ACCURACY, COMPLETENESS, VALIDITY OR RELEVANCE OF THE INFORMATION OR WHETHER OR NOT SUCH RIGHTS ARE ESSENTIAL OR NON-ESSENTIAL.

THE OPEN MOBILE ALLIANCE IS NOT LIABLE FOR AND HEREBY DISCLAIMS ANY DIRECT, INDIRECT, PUNITIVE, SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR EXEMPLARY DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF DOCUMENTS AND THE INFORMATION CONTAINED IN THE DOCUMENTS.

© 2003 Open Mobile Alliance Ltd. All Rights Reserved.

Used with the permission of the Open Mobile Alliance Ltd. under the terms set forth above.

Contents

- 1. SCOPE.....2
- 2. REFERENCES2
 - 2.1 NORMATIVE REFERENCES.....2
 - 2.2 INFORMATIVE REFERENCES.....2
- 3. TERMINOLOGY AND CONVENTIONS.....2
 - 3.1 CONVENTIONS.....2
 - 3.2 DEFINITIONS.....2
 - 3.3 ABBREVIATIONS2
- 4. INTRODUCTION2
 - 4.1 GOALS2
- 5. DRM CONTENT FORMAT2
 - 5.1 ISO BASE MEDIA FILE FORMAT2
 - 5.1.1 File structure2
 - 5.1.2 File Branding2
 - 5.2 COMMON BOXES.....2
 - 5.2.1 The Common Headers Box.....2
 - 5.2.2 Extended Headers2
 - 5.3 DISCRETE MEDIA FORMAT.....2
 - 5.3.1 DCF MIME Type.....2
 - 5.3.2 DCF File Format.....2
 - 5.3.3 Overall structure.....2
 - 5.3.4 Multiple OMA DRM Containers2
 - 5.3.5 Metadata Support.....2
 - 5.4 PACKETIZED MEDIA FORMAT (PDCF).....2
 - 5.4.1 PDCF MIME Type2
 - 5.4.2 PDCF File format.....2
 - 5.4.3 PDCF Streaming format2
- APPENDIX A. STATIC CONFORMANCE REQUIREMENTS (NORMATIVE).....2
- APPENDIX B. RESERVED NUMBERS (INFORMATIVE)2
- APPENDIX C. CHANGE HISTORY (INFORMATIVE).....2

Figures

- Figure 1: DCF file header and body.....2
- Figure 3: DCF structure.....2
- Figure 5: Encrypted wrapper payload format.....2

Tables

- Table 1. Algorithm-id values.....2
- Table 2. PaddingScheme values.....2
- Table 3: Logical DCF box structure diagram2
- Table 4. OMA DRM discrete media header fields2
- Table 5: Content Object box.....2

Table 6: Encrypted Payload Wrapper fields2

Table 7: Required OMA DRM specific parameters2

Table 8: Reserved identifier constants in the DCF format.....2

Table 9: Reserved OMA DRM specific identifier constants in the PDCF format2

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.

A number of DRM specifications have already been defined within the OMA. See [DRM], [DRMCF] and [DRMREL]. These existing specifications are referred to within this document as “release 1”.

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 “*Digital Rights Management*” specification [DRM-v2].

2. References

2.1 Normative References

- [CREQ] “Specification of WAP Conformance Requirements”, Open Mobile Alliance™, WAP-221-CREQ. <http://www.openmobilealliance.org/>
- [DRM] “Digital Rights Management”, Open Mobile Alliance™. OMA-Download-DRM-v1_0-20020905-CDRM v1
- [DRMCF] ”DRM Content Format”, Open Mobile Alliance™, OMA-DRM-DRMCF-v1_0
- [DRMREL] “DRM Rights Expression Language”. Open Mobile Alliance™. OMA-DRM-DRMREL-v2_0. <http://www.openmobilealliance.org/>
- [DRM-v2] “Digital Rights Management”. Open Mobile Alliance™. OMA-DRM-DRM-v2_0. <http://www.openmobilealliance.org/>
- [ISO14496-12] “Information technology — Coding of audio-visual objects – Part 12: ISO Base Media File Format”, International Organisation for Standardisation, ISO/IEC 14496-12, 2003
- [ISO7498-2] **TODO**
- [RFC2119] “Key words for use in RFCs to Indicate Requirement Levels”. S. Bradner. March 1997. [URL:http://www.ietf.org/rfc/rfc2119.txt](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](http://www.ietf.org/rfc/rfc2234.txt)
- [RFC2392] “Content-ID and Message-ID Uniform Resource Locators”. E. Levinson. August 1998. <http://www.ietf.org/rfc/rfc2392.txt>
- [RFC2396] ”Uniform Resource Identifiers (URI): Generic Syntax”, T. Berners-Lee et al. August 1998, <http://www.ietf.org/rfc/rfc2396.txt>
- [RFC2616] “Hypertext Transfer Protocol -- HTTP/1.1”. R. Fielding, et al. June 1999. <http://www.ietf.org/rfc/rfc2616.txt> .
- [RFC2630] “Cryptographic Message Syntax”. R. Housley. June 1999. <http://www.ietf.org/rfc/rfc2630.txt>
- [TS26.234] “Transparent end-to-end packet-switched streaming service (PSS); Protocols and Codecs”, The Third Generation Partnership Project, TS-26.234
- [TS26.244] “Transparent end-to-end pPacket-switched sStreaming sService (PSS); File Format”, The Third Generation Partnership Project, TS-26.244
- [WSP] "Wireless Session Protocol". WAP Forum™. WAP-230-WSP. <http://www.openmobilealliance.org/>

2.2 Informative References

- [WAPARCH] “WAP Architecture”. Open Mobile Alliance™. WAP-210-WAPArch. [URL:http://www.wapforum.org/](http://www.wapforum.org/)

<<If there are no references of a particular type, state that there are none>>

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

Asset	Content governed by rights. See DRM content.
Box	A binary data structure conforming to elementary data type definitions in [ISO14496-12]
Composite object	A content object that contains one or more Media Objects by means of inclusion.
Confidentiality	The property that information is not made available or disclosed to unauthorised individuals, entities or processes. (From [ISO7498-2])
Content	One or more Media Objects
Content Issuer	The entity making content available to the DRM Agent in a device.
Content Provider	An entity that is either a Content Issuer or a Rights Issuer.
Content Retailer	An entity that is a Content Issuer and/or a Rights Issuer.
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	Content that is consumed according to a set of rights. DRM content may be in encrypted DRM Content Format or in plaintext delivered inside a DRM message
DRM Message	An OMA DRM Release 1 term defined in [DRM]
Integrity	The property that data has not been altered or destroyed in an unauthorised manner. (ISO7498-2)
Media object	A digital resource e.g. a ringing tone, a screen saver, a Java game or a composite object.
Media type	A MIME media type.
Permission	Actual usages or activities allowed (by the Rights Issuer) over Protected Content (From [ODRL1.1])
Play	To create a transient, perceivable rendition of a resource (From [MPEG21 RDD])
Protected Content	Media Objects that are consumed according to a set of Permissions in a Rights Object.
Rights	Permissions and constraints defining under which circumstances access is granted to DRM content.
Rights issuer	An entity who issues rights objects.
Rights Issuer	An entity that issues Rights Objects to OMA DRM Conformant Devices.
Rights Object	A collection of Permissions and other attributes which are linked to Protected Content.

Rights Object Acquisition Protocol (ROAP)	A protocol defined within this specification. This protocol enables devices to request and acquire Rights Objects from a Rights Issuer.
Superdistribution	A mechanism that (1) allows a User to distribute Protected Content to other Devices through potentially insecure channels and (2) enables the User of that Device to obtain a Rights Object for the superdistributed Protected Content.
User	The human user of a Device. The User does not necessarily own the Device.

3.3 Abbreviations

3GPP	3rd Generation Partnership Project
4CC	Four Character Code
AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
CEK	Content Encryption Key
CI	Content Issuer
CTR	Counter Mode
DCF	DRM Content Format
DRM	Digital Rights Management
HTTP	Hypertext Transfer Protocol
ISO	International Standards Organization
MIME	Multipurpose Internet Mail Extensions
MMS	Multimedia Messaging Service
OMA	Open Mobile Alliance
PDCF	Packetized DRM Content Format
PSS	Packet switched Streaming Service
RFC	Request For Comments
RI	Rights Issuer
RO	Rights Object
ROAP	Rights Object Acquisition Protocol
RTP	Real time Transport Protocol
RTSP	Real Time Streaming Protocol
SMS	Short Messaging Service
URI	Uniform Resource Indicator
URL	Uniform Resource Locator
URN	Uniform Resource Name

4. Introduction

OMA Digital Rights Management defines a delivery method in which the Media Object is encrypted and the Rights containing the encryption key are delivered to the Device apart from the Media Object. This specification defines the DRM Content Format for encrypted Media Objects.

The DRM Content Format is closely related to the Rights Expression Language specification [DRMREL], which defines the syntax and semantics for the Rights Objects.

4.1 Goals

In addition to encrypting the Media Object the DRM 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 later while maintaining compatibility with the older versions. Compatibility with the version 1 Content Format [DRMCF] is not maintained by this specification, thus the MIME type shall be changed as well.

There are two profiles of the 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 data structures for the purpose of reusing components. 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 CI 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 DCF looks like a DRM protected file, whereas a PDCF looks and functions like a media file to the outside.

5. DRM Content Format

This section defines the DRM Content Format for Protected Content.

There are two DRM Content Format profiles:

- DCF: The first profile is used 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]. By using the ISO principles, the DCF format maintains the extensible nature of the ISO format, while keeping overhead minimal. An OMA DRM Device defined in [DRM-v2] MUST support the DCF format as defined in this specification. In addition, version 1 DCF as defined in [DRMCF] MAY be supported.
- PDCF: The second profile is used to protect Continuous (packetized) Media (i.e. Audio and Video.) Continuous media is protected in a separate format because it is packetized. 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 packetization is also required in order to stream continuous media. An OMA DRM compliant streaming server MUST be able to understand the Protected Format's structure in order to break the content into headers and packets that can be delivered to a client that understands the Protected Format.

5.1 ISO Base Media File Format

The Discrete Media profile (DCF) is based on the ISO Base Media File Format data types and conventions as defined in [ISO14496-12]. 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 also based on the ISO base media file format, but is defined in a separate specification, in the 3GPP [TS26.244]. 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 File structure

The ISO base media file format is structured around an object-oriented design of boxes. A basic box has two mandatory fields, length and type. The type identifier is used to dynamically bind a box to a statically defined type and the length is an implicit offset to the end of the box. A Box type identifier is a *Unique Identifier Number*. List of reserved numbers can be found in Appendix B. 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.

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;
}
```

In files conforming to this specification, box *size* MUST be greater than 1 (e.g. *largesize* in the ISO specification not used) and the *extended_type* MUST NOT be used in the mandatory boxes. Also note that in some earlier ISO specifications, the term *atom* was used to describe the file format structures, but the data structures specified in this specification SHALL be called *boxes* in order to be consistent with 3GPP and current ISO specifications.

Box alignment is by default to the next byte boundary in the end of the box. Extra padding should not be needed as all datatypes in 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;
}
```

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. In any case, a conforming parser MUST support checking the version number field.

A representation of the FullBox above is:

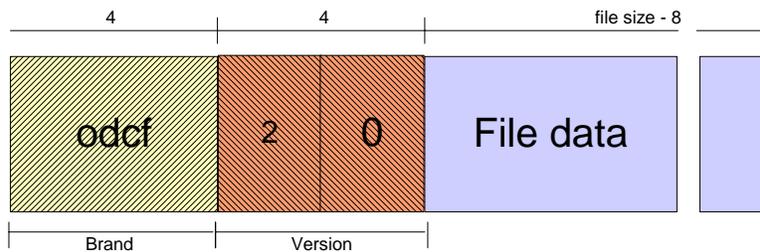
Name	Type	Value
Size	Unsigned int(32)	Offset to the end of the box
Type	Unsigned int(32)	Box type 4CC
Version	Unsigned int(8)	Version field
Flags	Unsigned int(24)	Additional flags

All numeric fields in the file format MUST be in network byte order.

5.1.2 File Branding

The ISO base media file format allows for a file signature/brand in the file header. Files conforming to the Discrete Media profile MUST include a brand number. The file brand is 32 bits (4 octets) wide with the hexadecimal value 0x6F646366 ('odcf'). This MUST be followed by a four-octet version indicator, making the file brand a total of eight octets (64 bits) from the beginning of the file. The version field consists of a version major and minor numbers, two octets each, in network order. For files conforming to this version of the DCF specification the version value MUST be 2.0 (0x00020000). A conforming file parser MUST support the version number. The Figure 1 shows the relationship of the file brand, version and rest of the file content.

Figure 1: DCF file header and body



The ISO file type box 'ftyp' MUST NOT be used in a version 2.0 DCF due to its variable size. Files conforming to the Continuous Media profile (PDCF) MUST include a file type box as specified in [TS26.244].

5.2 Common Boxes

5.2.1 The Common Headers Box

```
aligned(8) class OMADRMCommonHeaders extends FullBox('odhe', version, 0) {
    unsigned int(16)    EncryptionMethod;    // Encryption method
    unsigned int(16)    EncryptionPadding;   // Padding type
    unsigned int(32)    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[];  // Extensible headers, to the end of the box
}
```

The Common Headers box defines a structure for the required headers. 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 5.3.3.2 and 5.4.2.2.5 for details.

A Device **SHOULD NOT** edit 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 below.

Table 1. Algorithm-id values

Algorithm-id	Value	Semantics
NULL	0x0000	No encryption for this object
AES_128_CBC	0x0001	AES symmetric encryption as defined by NIST. 128 bit keys. Cipher block chaining mode (CBC). 128 bit initialization vector prefixing the ciphertext. Padding according to RFC 2630, unless overridden by the <i>PaddingScheme</i> field.
AES_128_CTR	0x0002	AES symmetric encryption as defined by NIST. 128 bit keys. Counter mode (CTR). 128 bit IV is constructed using a unique counter that prefixes the ciphertext.

Rights Issuers MUST take care in using NULL EncryptionMethod because, given a null-encrypted element within a DCF, the following statements hold true:

- Null-encrypted elements do not have any Confidentiality protection.
- Null-encrypted elements can be used without an associated Rights Object.
- Null-encrypted elements may not have any integrity protection, because the hash for integrity check is included in the associated Rights Object.

5.2.1.2.1 PaddingScheme Field

The *PaddingScheme* parameter defines how the last block of ciphertext is padded.

Values of the *PaddingScheme* field are defined in the table below:

Table 2. PaddingScheme values

Padding-Scheme	Value	Semantics
NULL	0x0000	No padding. This padding-scheme MUST only be used if the <i>PlaintextLength</i> parameter is greater than zero.
RFC_2630	0x0001	Padding according to RFC 2630. If this padding scheme is used, <i>PlaintextLength</i> MUST be zero.

5.2.1.3 PlaintextLength Field

The *PlaintextLength* field defines the length of the original plaintext. Some simple padding schemes may require that the plaintext length is explicitly defined. If the field is not used, it MUST be set to zero.

5.2.1.4 ContentIDLength Field

The *ContentIDLength* field defines the number of bytes occupied by the *ContentID* field.

5.2.1.5 RightsIssuerURLLength Field

The *RightsIssuerURLLength* field indicates the number of bytes occupied by the *RightsIssuerURL* field.

5.2.1.6 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.

5.2.1.7 ContentID Field

The *ContentID* field MUST contain a unique identifier for this DRM protected content object. The value MUST be associated with a CEK. The value MUST be encoded using US-ASCII encoding.

The value MUST be a URI according to [RFC2396]. It is the responsibility of the content author to guarantee the uniqueness of the *ContentID*. URI schemes like “cid:local-part@domain” as defined in [RFC2392] MAY be used.

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].

5.2.1.8 RightsIssuerURL Field

The *RightsIssuerURL* field defines the Rights Issuer URL. The Rights Issuer URLs MAY be used by the consuming device to obtain rights for this DRM protected content object. The mechanism is defined in OMA DRM specification [DRM-v2]. 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].

5.2.2 Extended Headers

There are two mechanisms to extend the mandatory header information. The *TextualHeaders* and *ExtendedHeaders* fields 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 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 of 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 extended 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 extended textual headers:

```
Content-Vendor:GreatCompany\0Icon-URI:http://www.greatcompany.com:8080/contenticon.png\0
```

Each supported header is defined using augmented Backus-Naur Form (BNF) [RFC2234]. The extended headers are encoded using UTF-8 encoding.

The *ExtendedHeaders* array is used for future additions, and it can nest e.g. binary data. The array MAY have zero or more sub-boxes, and it spans until the end of the *OMADRMCommonHeaders* box is reached. The *TextualHeaders* field SHOULD NOT be used for large strings, such as encoded binary data, the *ExtendedHeaders* field SHOULD be used instead. A Device MAY ignore the extended headers it does not support.

5.2.2.1 Silent header

The *Silent* header is an indication to the client that the Rights Object for this DCF can be obtained silently from the Rights Issuer, without user interaction for payments, etc. The device MAY use this header to determine how to acquire the Rights Object.

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

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, opportunistically.

The parameter *silent-rights-url* MUST be a URL according to [RFC2396].

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.

If this request cannot be reconciled to a prior purchase transaction, the RI server MUST return an error. The client can take further action based on this error indication. It is recommended that the client start a browsing session with the RI URL if the context is a user-initiated session. If the context is a DRM-agent initiated session to acquire rights silently and automatically, then it is better for the client to abandon the rights acquisition effort.

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 DCF.

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-method	Semantics
“instant”	This indicates that one of the elements within this composite object can be used for preview. If <code>instant</code> method is specified, then <code>preview-element-uri</code> 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 <code>preview-rights</code> method is specified, then <code>preview-rights-url</code> 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 DCF.

The parameter `preview-rights-url` MUST be a URL according to RFC2396.

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, the result is either a Rights Object or an error. This MUST NOT result in any re-direction.

5.2.2.3 ContentName header

The *ContentName* header contains a descriptive name for this DRM protected content object. The name is only informative and the device MAY use it e.g. to derive a filename when the DRM protected object is received and stored into a local repository. Other names may be transmitted outside this object (e.g. `Content-Disposition` header in HTTP) and they may override the name specified in this element.

```
ContentName := "Content-Name" ":" token
```

5.2.2.4 ContentDescription header

The *ContentDescription* header contains a description of the DRM protected content object. This text is informative and the device MAY display it to the user prior to using the `RightsIssuer` field.

```
ContentDescription := "Content-Description" ":" token
```

5.2.2.5 ContentVendor header

The *ContentVendor* header contains a textual string representing the name of the organisation that provided the media object. This text is informative and the device MAY display it to the user prior to using the Rights Issuer URL field.

```
ContentVendor := "Content-Vendor" ":" token
```

5.2.2.6 IconURI header

The *IconURI* header contains a URI where an appropriate icon for this content may be retrievable from. The device MAY use this header to 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.

```
IconURI := "Icon-URI" ":" token
```

5.2.2.7 Unsupported headers

Content author MAY insert additional headers to the *TextualHeaders* field. Additional 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.

5.3 Discrete Media Format

5.3.1 DCF MIME Type

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

```
application/vnd.oma.drm.content.v2?
```

5.3.2 DCF File Format

The structure of the Discrete Media profile of DRM protected content (DCF) MUST be according to the structure definitions below. The file brand of eight octets MUST precede the first box.

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 brand, and they MUST all be on the top level in the nesting structure.

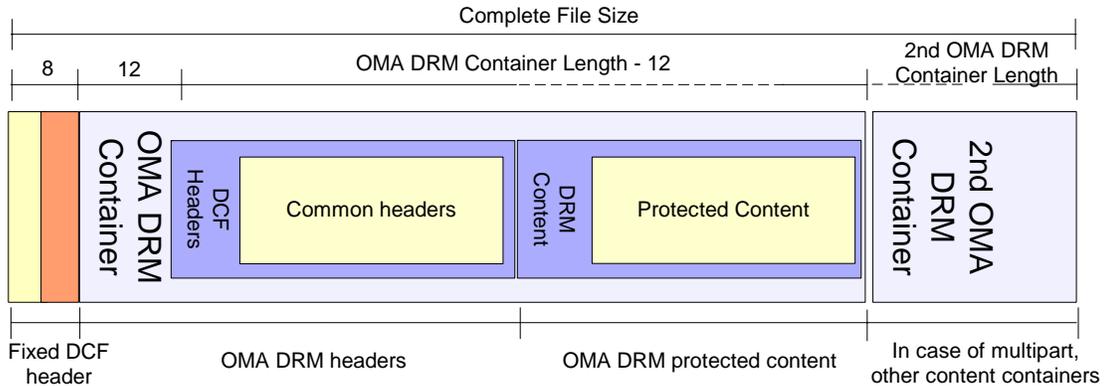
The *version* indicator field in each box MUST be 0 for files conforming to this specification.

5.3.3 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 with Brand number and 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 eight octets and the DCF headers box MUST be the first box in the OMA DRM Container.

Figure 2: DCF structure



The table below outlines the mandatory boxes and their order. Additional boxes MAY be added after the mandatory boxes have first appeared. Table 3 shows the nesting order of the mandatory boxes, on the left is the parent and on the right, the child.

Table 3: Logical DCF box structure diagram

Data type/value	Nesting level	Offset from beginning of file	Field purpose		
'odcf'	0	0	File header magic (4 bytes)		
0x00020000	0	4	File version (2 bytes major, 2 bytes minor)		
Box('odrm')	0	8	OMA DRM container box		
	Box('odhe')	1	20	DCF headers box	
		Box('ohdr')	2	36 + ContentTypeLength	OMA DRM common headers box
	Box('odda')	1	20 + Box('odhe')	OMA DRM content data box	
Box('odrm')	0	8 + Box('odrm')		If multipart DCF, additional OMA DRM container box	

5.3.3.1 OMA DRM Container Box

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

The OMADRMContainer box MUST include a single OMADRMHeaders box and a single OMADRMContent box, followed by optional extensions. The Extensions inside the OMADRMContainer box are defined by OMA.

5.3.3.2 Discrete Media Headers Box

```
aligned(8) class OMADRMDiscreteHeaders extends FullBox('odhe', version, 0) {
    unsigned int(8)      ContentTypeLength;    // Content Type Length
    char               ContentType[];         // Content Type String
    OMADRMCCommonHeaders CommonHeaders;      // Common headers (same as with PDCF)
}
```

The Discrete Media profile headers box includes fields specific to the DCF format and the common headers 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. In addition, the discrete headers box includes the common headers box. There MUST be exactly one OMADRMCCommonHeaders (see section 5.2.1 for details) box per a single OMADRMDiscreteHeaders box.

Table 4. 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

5.3.3.2.1 ContentType

The *ContentType* field MUST indicate the original MIME media type of the DRM protected content i.e. what content type the result of a successful decryption of the OMADRMCContent box represents. The *ContentType* field is encoded using US-ASCII encoding and MUST NOT include a NULL character.

5.3.3.2.2 CommonHeaders

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

5.3.3.3 Content Object Box

```
aligned(8) class OMADRMCContentObject extends FullBox('odda', version, 0) {
    unsigned int(32) OMADRMDDataLength;    // Length of the encrypted content
    bit(8) OMADRMDData[];                 // Encrypted content
}
```

The Content Object box MUST include only the data length field and data bytes for a single Protected Content Object. Later revisions of this box may include additional fields, so conforming implementations MUST use the OMADRMDDataLength field to indicate/determine the amount of actual data bytes.

Table 5: Content Object box

Field name	Type	Purpose
OMADRMDDataLength	Unsigned int(32)	Length of the OMADRMDData field, in octets
OMADRMDData	bit(8) []	Protected Content bytes, as specified by the OMADRMDiscreteHeaders box

5.3.3.4 Extended Boxes

Any additional boxes contained in a single OMA DRM container box have not been defined yet.

5.3.4 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).

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

Note that a Multipart DCF is different from a DCF including a Composite Object. For Composite Objects (such as MIME multipart, ZIP and so on), there exists only one set of OMA DRM headers, and only one Rights Object, whereas for Multipart DCFs, there are separate headers for each object in the Multipart DCF, and support for having different Rights for Content Objects.

5.3.5 Metadata Support

Additional proprietary 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.

An example of metadata could be adding a box for ID3 tag or an RDF document to describe the content. The identifiers or exact content are not specified in this specification.

TODO: should we define a common format for media-specific metadata box? Like Box('meta')

5.4 Packetized Media Format (PDCF)

The Continuous (Packetized) Media profile is targeted for media content like audio and video. Audio and video files MAY be included in a DCF format, but for creating a consistent user experience for OMA DRM, the PDCF format SHOULD be used for continuous media types like audio and video.

5.4.1 PDCF MIME Type

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

video/3gpp or audio/3gpp

The internal file branding and structure must conform to the specification [TS26.244]. The PDCF format MAY be used for downloaded content or for hosting streamable content. The format is limited to the media types listed in the 3GPP specification.

An audio/3gpp file is an instance of a video/3gpp file containing only audio tracks. This specification will support both, but for clarity, only use video/3gpp to refer to the 3GPP media file format [TS26.244].

5.4.2 PDCF File format

The PDCF file format is a sub profile of the video/3gpp format, which is used for (downloaded) protected media content. The structure and conformance to the profile are not defined in this specification, but instead, this specification defines the OMA DRM key management part of the format. The video/3gpp file format allocates space for a “black box” describing the key management governing access to the media content. In a PDCF file, this box MUST be the OMADRMKMSBox.

The protected video/3gpp file format data structures are defined by the 3GPP, but this specification gives an overview of the data structures and presents their OMA DRM aspects. Other DRM mechanisms MAY be used in video/3gpp files supporting DRM, but not in PDCF files, as explained in this specification.

5.4.2.1 Original Sample Entries

Each packetized track in a PDCF file has a corresponding *Sample Entry*. The Sample Entry includes information about the bitstream, such as what resolution, codec etc. were used to make the encoding transform to the track, and how to make a reverse transform to reconstruct e.g. video to the display.

The original sample entries describing the track are replaced with a derived type, which hides the codec used to encode the corresponding track. This is done to make the sample format incompatible with non DRM aware Devices, and thus avoid any bad consequences, such as crashing the media player, while opening the media. A `ProtectionInfoBox` is appended to the derived sample entry and the extended sample entry box only indicates that an encrypting “codec” was used to encode the content. The actual codec, along with DRM specific parameters, is indicated in the `ProtectionInfoBox` and its child boxes.

For example, a video track sample entry is derived from the `VisualSampleEntry`, and the codec type is replaced with an encryption indicator ‘encv’. The `ProtectionInfoBox` containing the original codec identifier is appended to the box.

```
class EncVisualSampleEntry(codingname) extends VisualSampleEntry ('encv'){
    ProtectionInfoBox(codingname) info;
}
```

5.4.2.2 Protection Information

The `ProtectionInfoBox` is used in video/3gpp files to indicate that a track is DRM protected, and to carry information about the protection scheme. The `ProtectionInfoBox` is a container box.

The `OriginalFormatBox` is used to indicate the actual codec used, `SchemeTypeBox` to indicate the DRM key management system and `SchemeInformationBox` for passing key management system specific information. All of these boxes **MUST** appear in a `ProtectionInfoBox` as below, unless otherwise specified in [TS26.244].

```
aligned(8) class ProtectionInfoBox(fmt) extends FullBox('sinf', 0, 0) {
    OriginalFormatBox(fmt)          original-format;
    SchemeTypeBox                   scheme-type;
    SchemeInformationBox             info;
}
```

5.4.2.2.1 DRM Scheme Type

The `SchemeTypeBox` includes information on which DRM system is being used to manage keys and decryption of the content. A video/3gpp file **MAY** support also other key management systems than OMA DRM, the key management system in use is indicated by a 4CC in the *scheme_type* field.

For PDCF files conforming to this specification, the *scheme_type* **MUST** be the 4CC ‘odkm’, and *scheme_version* **MUST** be 0x0200 (version 2.0). If OMA DRM key management scheme ‘odkm’ is indicated, then the video/3gpp file is a PDCF and **MUST** contain at least one `OMADRMKMSBox`. A PDCF **SHALL** support only OMA DRM for the key management system.

```
aligned(8) class SchemeTypeBox extends FullBox('schm', 0, flags) {
    unsigned int(32)          scheme_type;          // 4CC identifying the scheme
    unsigned int(16)         scheme_version;       // scheme version
    if (flags & 0x000001) {
        unsigned int(8)      scheme_uri[];        // browser uri
    }
}
```

5.4.2.2.2 Scheme Information

The `SchemeInformationBox` 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 box in the array.

```
aligned(8) class SchemeInformationBox extends FullBox('schi', 0, 0) {
    Box      scheme-specific-data[];
}
```

5.4.2.2.3 OMA DRM Key Management System

There MAY be several instances of the OMADRMKMSBox in a PDCF file, and one can appear either at the toplevel *movie box* or exactly one per each protected track. There MUST NOT be key management boxes in both movie level and track level. The exact locations for these boxes are defined in the 3GPP file format specification [TS26.244].

```
aligned(8) class OMADRMKMSBox extends FullBox('odkm', version, 0) {
    OMADRMSampleFormatBox    sample_format;
    OMADRMCommonHeaders      headers;
}
```

Contained in the OMADRMKMSBox there MUST be a single OMADRMSampleFormatBox and a single OMA DRMCommonHeaders box. The sample format box is used to indicate the format of the payload headers placed on media access units. The internal structure of the access units might not make any sense to a Device that is not OMA DRM aware, but an OMA DRM aware Device will look at the sample format box to know how to extract the protected content.

5.4.2.2.4 Sample Format

The *Sample Format* specifies the format for each access unit, more specifically the payload header type for OMA DRM protected content when used with a streaming protocol such as the 3GPP PSS [TS26.234].

```
aligned(8) class OMADRMSampleFormatBox extends FullBox('osfm', 0, 0) {
    bit(1)          selective-encryption;
    bit(7)          reserved;
    unsigned int(8) key-indicator-length;
    unsigned int(8) IV-length;
}
```

For more information on the access unit format, see section 5.4.3.1.

5.4.2.2.5 Common Headers

The Common headers box is exactly the same as defined in section 5.2.1.

5.4.3 PDCF Streaming format

Streaming PDCF content is leveraging the protected video/3gpp file format, and widely deployed standard streaming protocols. This specification uses the 3GPP PSS service protocols [TS26.234] as a reference, but the encrypted payload wrapper format MAY be used in any other streaming service using RTSP streaming, SDP signaling and RTP transport.

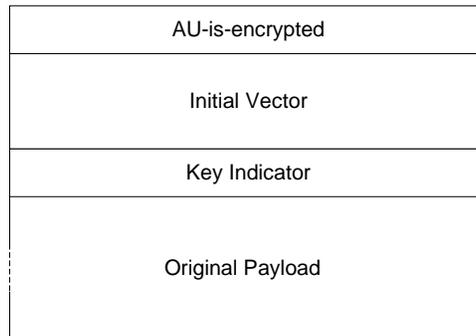
Supporting the PDCF streaming is OPTIONAL, even if PDCF format is supported. A multimedia streaming session MAY consist of PDCF streamed tracks and unprotected tracks.

Streaming protected tracks is signaled through SDP parameters, using information contained in the sample format entries of the protected video/3gpp file. A streaming server derives network packets from a *hint track* in the media file.

Although the streaming format and payload are defined in [TS26.244], they are explained here from the OMA DRM perspective.

5.4.3.1 RTP Payload

The RTP payload format consists of two parts: the encrypted payload wrapper and the actual media payload. The media payload (e.g. H.263 video) is packetized according to the appropriate standard. The encrypted payload wrapper includes a header with additional signaling 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.

Figure 3: Encrypted wrapper payload format

The DRM sample format in section 5.4.2.2.4 is used to signal the format for the encrypted payload wrapper. The fields in the wrapper format are prefixing the actual protected payload and to the RTP protocol layer, it only looks like it is transporting the wrapper payload media type.

Table 6: Encrypted Payload Wrapper fields

Parameter name	Purpose
AU-is-encrypted	Boolean encryption indicator (0=false, 1=true).
Initial Vector	IV for the access unit
Key Indicator	Key indicator for CTR mode

Add more details as 3GPP finalizes the payload format

5.4.3.2 Hint Tracks

The video/3gpp format supports special hint tracks for streaming servers. Hint tracks include pointers to network packets within the packetized file. Using these pointers, the streaming server is able to stream the file even without knowledge of what the packets actually include. Downloaded PDCFs MAY include hint tracks, but in normal playback of downloaded content, they are ignored.

5.4.3.3 Session signaling

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 parameters.

The generic parameters are defined in [TS26.234]. In the *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 zero (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 7: 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.

Appendix A. Static Conformance Requirements (Normative)

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

Item	Function	Reference	Status	Requirement
DRM-DCF-GEN-1	AES128CBC encryption algorithm	5.2.1.2	M	
DRM-DCF-GEN-2	RFC 2630 padding scheme	5.2.1.2.1	M	
DRM-DCF-GEN-3	NULL padding scheme	5.2.1.2.1	M	
DRM-DCF-GEN-4	PlaintextLength field	5.2.1.2.1	M	
DRM-DCF-GEN-5	RightsIssuer field	5.2.1.7	M	
DRM-DCF-GEN-6	ContentName header	5.2.2.3	O	
DRM-DCF-GEN-7	ContentDescription header	5.2.2.4	O	
DRM-DCF-GEN-8	ContentVendor header	5.2.2.5	O	
DRM-DCF-GEN-9	IconURI header	5.2.2.6	O	
DRM-DCF-GEN-10	Ignore unsupported headers	5.2.2.7	M	
DRM-DCF-GEN-11	AES128CTR mode encryption algorithm	5.2.1.2	O	Mandatory if PDCF is supported? Or is this always used for PDCFs?
DRM-DCF-GEN-12	DCF support	5.3	M	
DRM-DCF-GEN-13	PDCF support	5.4	O	
DRM-DCF-GEN-14	Ignore unsupported boxes in DCF		M	
DRM-DCF-GEN-15	Check DCF brand	5.1.2	M	
DRM-DCF-GEN-16	DCF version		M	
DRM-DCF-GEN-17	FullBox version	5.1.1	M	
DRM-DCF-GEN-18	64 bit box length	5.1.1	?	
DRM-DCF-GEN-19	PDCF streaming	5.4.3	O	

Appendix B. Reserved Numbers (Informative)

Table 8: Reserved identifier constants in the DCF format

UUID	Reference	Purpose
'odcf'	1.1.1	File brand
'odrm'	5.3.3.1	OMA DRM Container box
'ohdr'	5.2.1	Common headers box
'odhe'	5.3.3.2	Headers box for the Discrete Media profile box
'odda'	5.3.3.3	Protected Content box

Table 9: Reserved OMA DRM specific identifier constants in the PDCF format

UUID	Reference	Purpose
'odkm'	5.4.2.2.2, 5.4.2.2.3	OMA DRM scheme type, OMA DRM scheme information box identifier
'ohdr'	0	Common headers box

Appendix C. Change History (Informative)

A.1. Approved Version History

Reference	Date	Description
n/a	n/a	No prior version –or- No previous version within OMA

A.2. Draft/Candidate Version <current version> History

Document Identifier	Date	Sections	Description
Draft Versions OMA-DRM-DCF-V2_0	01 Oct 2003	Initial draft	Moved to new format and incorporated input from Sami
	03 Nov 2003		Removed DCF legacy version, added support for binary headers, major update of the PDCF sections
Candidate Version OMA-DRM-DCF-V1_2		n/a	Status changed to Candidate by TP TP ref # OMA-TP-2003-0abc-CandidateRequest_xxyz_V1_2

