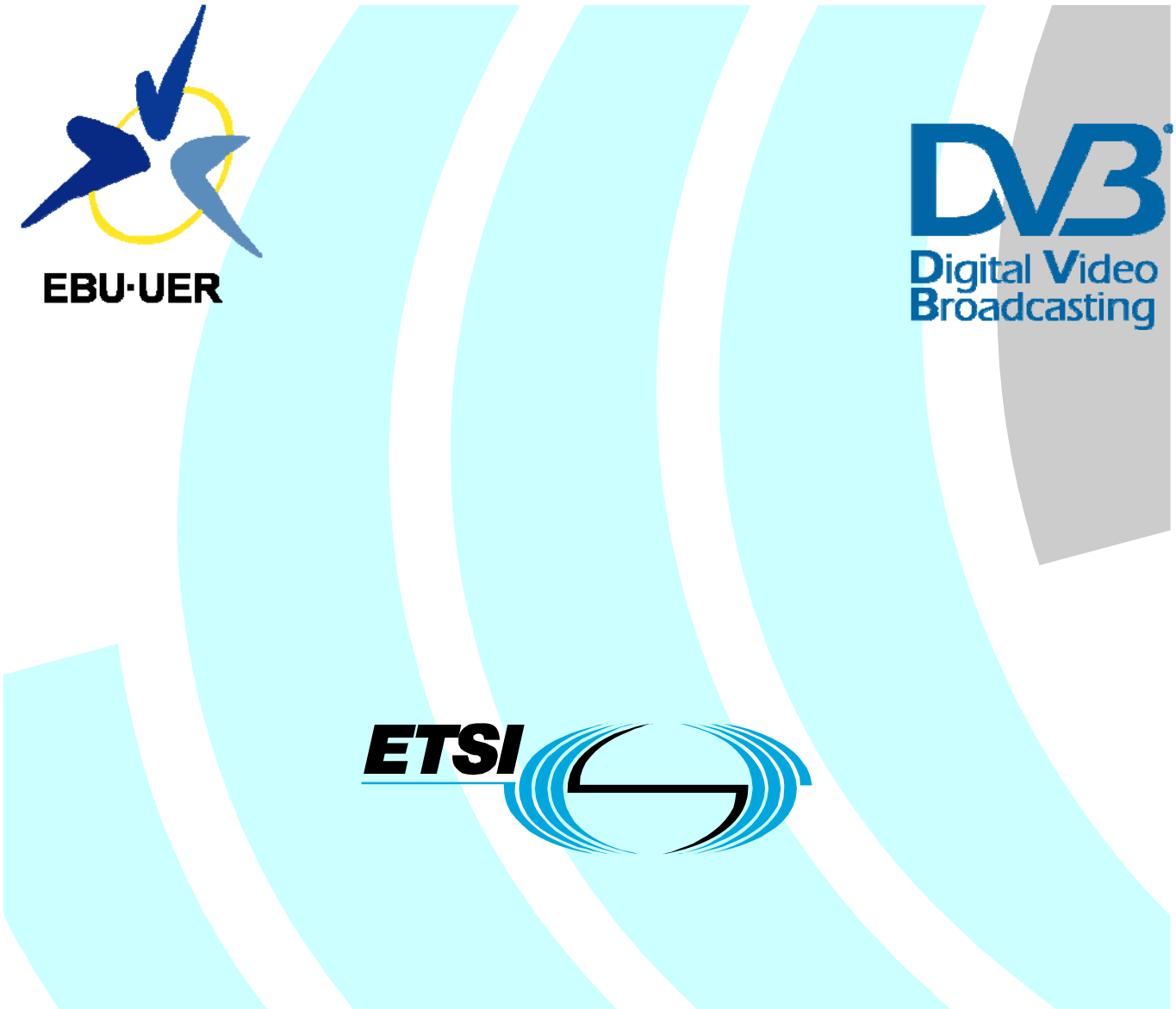
ETSI TR 102 825-12 V1.1.1 (2011-03)

Technical Report

Digital Video Broadcasting (DVB);
Content Protection and Copy Management (DVB-CPCM);
Part 12: CPCM Implementation Guidelines



Reference
DTR/JTC-DVB-222-12

Keywords
broadcast, DVB

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

Content

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Report (TR) has been produced by Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI).

NOTE: The EBU/ETSI JTC Broadcast was established in 1990 to co-ordinate the drafting of standards in the specific field of broadcasting and related fields. Since 1995 the JTC Broadcast became a tripartite body by including in the Memorandum of Understanding also CENELEC, which is responsible for the standardization of radio and television receivers. The EBU is a professional association of broadcasting organizations whose work includes the co-ordination of its members' activities in the technical, legal, programme-making and programme-exchange domains. The EBU has active members in about 60 countries in the European broadcasting area; its headquarters is in Geneva.

European Broadcasting Union
CH-1218 GRAND SACONNEX (Geneva)
Switzerland
Tel: +41 22 717 21 11
Fax: +41 22 717 24 81

The Digital Video Broadcasting Project (DVB) is an industry-led consortium of broadcasters, manufacturers, network operators, software developers, regulatory bodies, content owners and others committed to designing global standards for the delivery of digital television and data services. DVB fosters market driven solutions that meet the needs and economic circumstances of broadcast industry stakeholders and consumers. DVB standards cover all aspects of digital television from transmission through interfacing, conditional access and interactivity for digital video, audio and data. The consortium came together in 1993 to provide global standardisation, interoperability and future proof specifications.

The present document is part 12 of a multi-part deliverable. Full details of the entire series can be found in part 1 [i.3].

Introduction

CPCM is a system for Content Protection & Copy Management of commercial digital content delivered to consumer products. CPCM manages content usage from acquisition into the CPCM system until final consumption, or export from the CPCM system, in accordance with the particular usage rules of that content. Possible sources for commercial digital content include broadcast (e.g. cable, satellite, and terrestrial), Internet-based services, packaged media, and mobile services, among others. CPCM is intended for use in protecting all types of content - audio, video and associated applications and data. CPCM provides specifications to facilitate interoperability of such content after acquisition into CPCM by networked consumer devices for both home networking and remote access.

This first phase of the specification addresses CPCM for digital Content encoded and transported by linear transport systems in accordance with TS 101 154 [i.1]. A later second phase will address CPCM for Content encoded and transported by systems that are based upon Internet Protocols in accordance with TS 102 005 [i.2].

1 Scope

The present document provides the Implementation Guidelines for the Digital Video Broadcasting (DVB) Content Protection and Copy Management (CPCM) system.

Implementation guidelines include:

- Guidelines for implementers of CPCM Instances and Devices.
- Guidelines for those who will provide Content to CPCM.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

Not applicable.

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TS 101 154: "Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream".

[i.2] ETSI TS 102 005: "Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in DVB services delivered directly over IP protocols".

[i.3] ETSI TS 102 825-1: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 1: CPCM Abbreviations, Definitions and Terms".

[i.4] ETSI TS 102 825-2: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 2: CPCM Reference Model".

[i.5] ETSI TR 102 825-13: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 13: CPCM Compliance Framework".

[i.6] ETSI TS 102 825-14: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 14: CPCM Extensions".

[i.7] ETSI TS 102 825-3: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 3: CPCM Usage State Information".

[i.8] ETSI TR 102 825-11: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 11: CPCM Content management scenarios".

[i.9] ETSI TS 102 825-4: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 4: CPCM System Specification".

[i.10] ETSI EN 300 468 "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".

[i.11] ETSI TS 102 825-9: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 9: CPCM System Adaptation Layers".

[i.12] ETSI TS 102 825-7: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 7: CPCM Authorized Domain Management".

[i.13] ETSI TS 101 162: "Digital Video Broadcasting (DVB); Allocation of identifiers and codes for Digital Video Broadcasting (DVB) systems".

[i.14] ETSI TS 102 825-5: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 5: CPCM Security Toolbox".

[i.15] ETSI TS 102 833: "Digital Video Broadcasting (DVB); File Format Specification for the Storage and Playback of DVB Services".

[i.16] Wolfgang Killmann, Werner Schindler: "A Proposal for functionality classes and evaluation methodology for true (physical) random number generators." .

[i.17] NIST SP800-22: "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications".

[i.18] FIPS140-1: "Security Requirements for cryptographic modules.".

[i.19] FIPS140-2: "Security Requirements for cryptographic modules".

[i.20] G. Marsaglia: "DIEHARD Statistical Tests".

NOTE: Available at http://www.stat.fsu.edu/pub/diehard/.

[i.21] ETSI TS 101 211: "Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI)".

[i.22] ISO/IEC 14496-12:2008: "Information Technology - Coding of audio-visual objects - Part 12: ISO base media file format", third edition.".

[i.23] ETSI TS 102 905: "Digital Video Broadcasting (DVB); Technical Specification for DVB Services in the Home Network Phase 1".

[i.24] IETF RFC 791: "Internet Protocol (IP)".

[i.25] CENELEC EN 50221 "Common Interface Specification for Conditional Access and other Digital Video Broadcasting Decoder Applications.".

[i.26] ETSI TS 101 699 (V1.1.1): "Digital Video Broadcasting (DVB); Extensions to the Common Interface Specification".

[i.27] ETSI TS 102 034: "Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB Services over IP Based Networks".

[i.28] ETSI TS 102 825-10: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 10: CPCM Acquisition, Consumption and Export Mappings".

[i.29] ISO/IEC 7816: " Identification cards -- Integrated circuit(s) cards with contacts".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 102 825-1 [i.30] apply.

NOTE: In some cases, for ease of reading, definitions are repeated in the present document. In case of conflict, the terms in TS 102 825-1 [i.31] will take precedence.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 102 825-1 [i.32] and the following apply.

NOTE: In some cases, for ease of reading, abbreviations are repeated in the present document. In case of conflict, the abbreviation in TS 102 825-1 [i.33] will take precedence.

CI	Common Interface
PRNG	Pseudo Random Number Generator
TRNG	True Random Number Generator

4 Reference Model implementation

4.1 CPCM Device and CPCM Instances

4.1.1 Implementing different CPCM Instances in the same CPCM Device

A CPCM Device may implement one or more CPCM Instances.

The decision to have one or more CPCM Instances in one CPCM device is guided by the following considerations:

- A CPCM Instance may implement several CPCM Functions.
- Each CPCM Instance has its own certificate and as such may be individually identified or revoked. It is therefore possible to revoke a given CPCM Instance in a CPCM Device without affecting the behaviour of other CPCM Instances present in the same CPCM Device.
- As each CPCM Instance may be revoked separately, implementations need to be such that the compromising of one CPCM Instance does not affect other CPCM Instances in the same CPCM Device. Consequently, a CPCM Instance will not be able to access Secure Data of other CPCM Instances in the same CPCM Device. Similarly, implementations of two CPCM Instances in the same CPCM device will be distinct so that the compromising of one implementation does not affect the behaviour of other implementations. Thus, the implementation cost is likely to grow with the amount of different CPCM Instances.
- Unless otherwise authorised by the applicable C&R Regime, messaging (including message protection) between CPCM Instances within the same CPCM device is the same as messaging between CPCM Instances in different devices.

It is recommended to minimise whenever possible the number of CPCM Instances implemented in a same device, unless one of the above properties is respected.

Different CPCM Instances within a single device may comply with different C&R regimes. This can allow for a CPCM Device to comply with C&R regimes that do not trust each other, C&R regimes that do not have matching compliance rules, or C&R regimes that cannot deliver CPCM Instance Certificates with the relevant C&R regime flag set.

NOTE: Minimizing the number of CPCM Instances might not be possible on open platforms like Personal Computers where different CPCM Instances may run independently and without knowing each other (see clause 4.2.12).

4.1.2 CPCM Instances and CPCM Functions

There is not necessarily a one-to-one relationship between a CPCM Instance and a CPCM Function:

- A CPCM Instance may implement one or more CPCM Functions.
- An implementer may choose to implement a single CPCM Function using more than one hardware item, i.e. across different CPCM Instances. When an Export Point or a Consumption Point is split over different hardware items and the CPCM device is ADM capable, the C&R regime may decide to count for ADSE each of the involved CPCM Instance as 1 or to attribute the count to a single CPCM Instance.

Both cases are illustrated in clauses 4.1.2.1 and 4.1.2.2.

A CPCM Function type (i.e. AP, PE, EP, CP or SE) is considered to be logically present only once within a CPCM Instance, even if it fulfils more than one purpose.

EXAMPLE 1: A CPCM Instance that is able to Acquire content from two different CPS will be considered to implement one Acquisition Point rather than two.

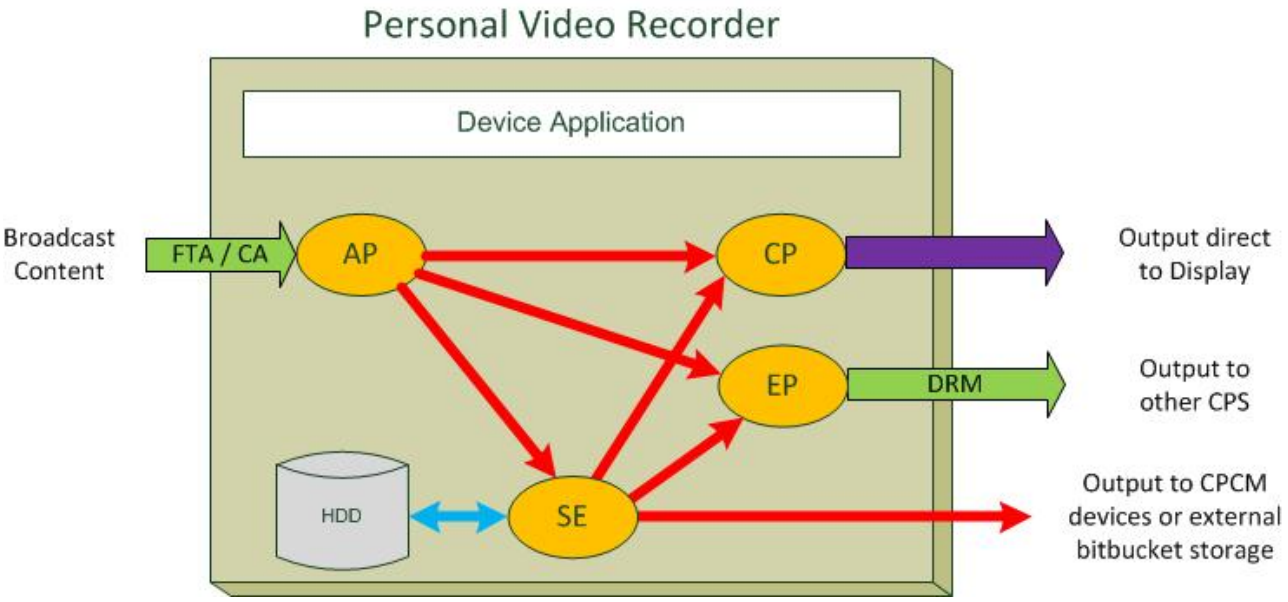
EXAMPLE 2: A Consumption Point that is able to Consume directly to a screen and through a protected link (e.g. HDCP) is considered as one Consumption Point.

The attention of the implementer is drawn to the fact that a C&R Regime may constrain the number of Functions implemented in a single CPCM Instance for a CICF and impose a count greater than one for CPCM Devices having a large number of CPCM Functions. Similarly, the choice to implement two Consumption Points or two Export Points in two different CPCM Instances could lead to a count of two as for ADSE instead of one dependent upon C&R regime.

4.1.2.1 CPCM Device Example

Figure 1 shows an example of a Personal Video Recorder. The associated CPCM Instance includes an Acquisition Point, a Storage Entity, an Export Point and a Consumption Point.

Even if the utility of CPCM may be not obvious in such a case, the attention of the implementer is drawn to the fact that any external storage, whether CPCM compliant or not, may be used in addition to the memory available in the device. This would not be achievable without the protection provided by CPCM. Furthermore, this device may exchange Content with other CPCM devices, when allowed by applicable USI.



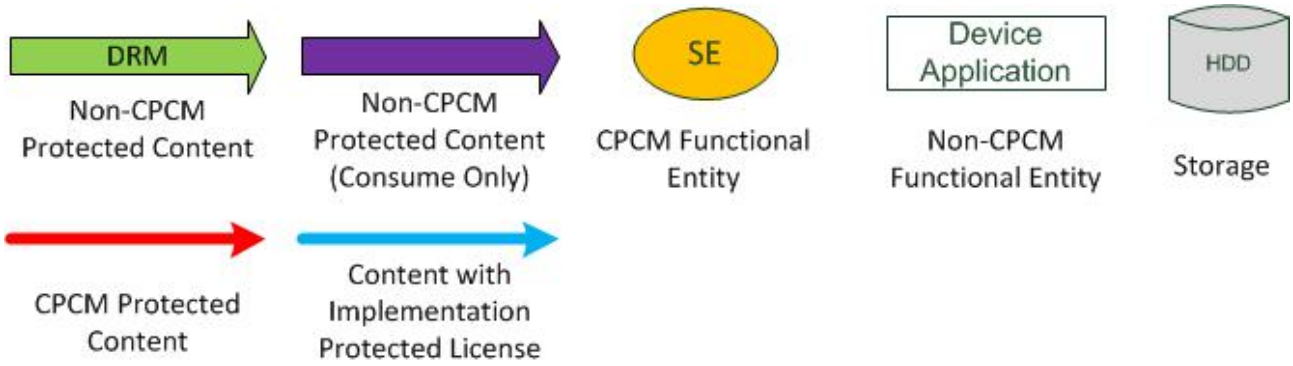


Figure 1: Example of Personal Video Recorder

4.1.2.2 CPCM Device with a smart card plug-in

TS 102 825-2 [i.4] gives different models of Acquisition Point architecture involving the use of a smart card to embed a CA system.

It is also possible to implement other CPCM functional entities using smart cards. Using smart cards can achieve better security both for software and key protection.

Figure 2 shows an example on how to implement a Consumption Point using a smart card.

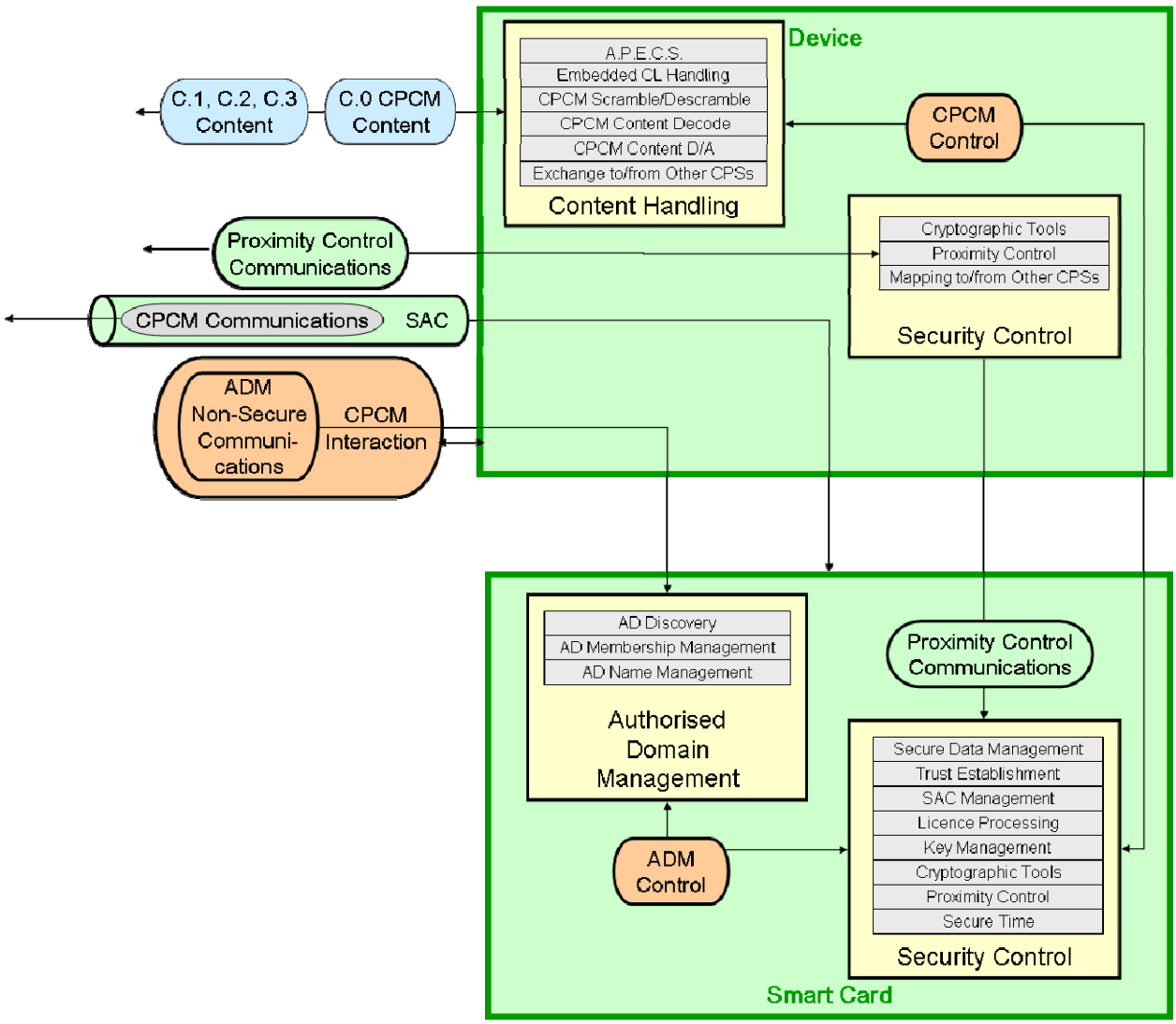


Figure 2: Smart card based implementation

In this case the device is to dealing with content handling, while the smart card deals with Authorised Domain management and security. Proximity Control is present in both the smart card and the device since it has to be performed between the CPCM device and other devices on one side and between the smart card and the CPCM device on the other side.

The interface between the smart card and the device needs to be secure. Several implementation options exist:

- Using Proprietary extension (and proprietary security); in this case, the smart card will work only with devices implementing that proprietary extension, most likely with devices from the same manufacturer.
- Using a standardised CPCM extension: in this case the interface will probably be secured using CPCM SAC. However, additional messages may need to be defined for the extension (e.g. to transmit the Content Descrambling Key). The smart card is likely to work with any device from any manufacturer, as long as it implements the CPCM extension.

4.2 Selection of suitable CPCM Functions for a device

This clause gives examples for CPCM implementers in order to help them determine which CPCM Functions need to be implemented in their products to achieve their design requirements.

4.2.1 Portable Media Player

This example of a Portable Media Player, PMP, is a device that is able to receive CPCM content from a CPCM sourcing device to record it locally and to play back the content for consumption. This playback can occur on the integrated screen or on a remote display using a digital link, protected with a dedicated CPS.

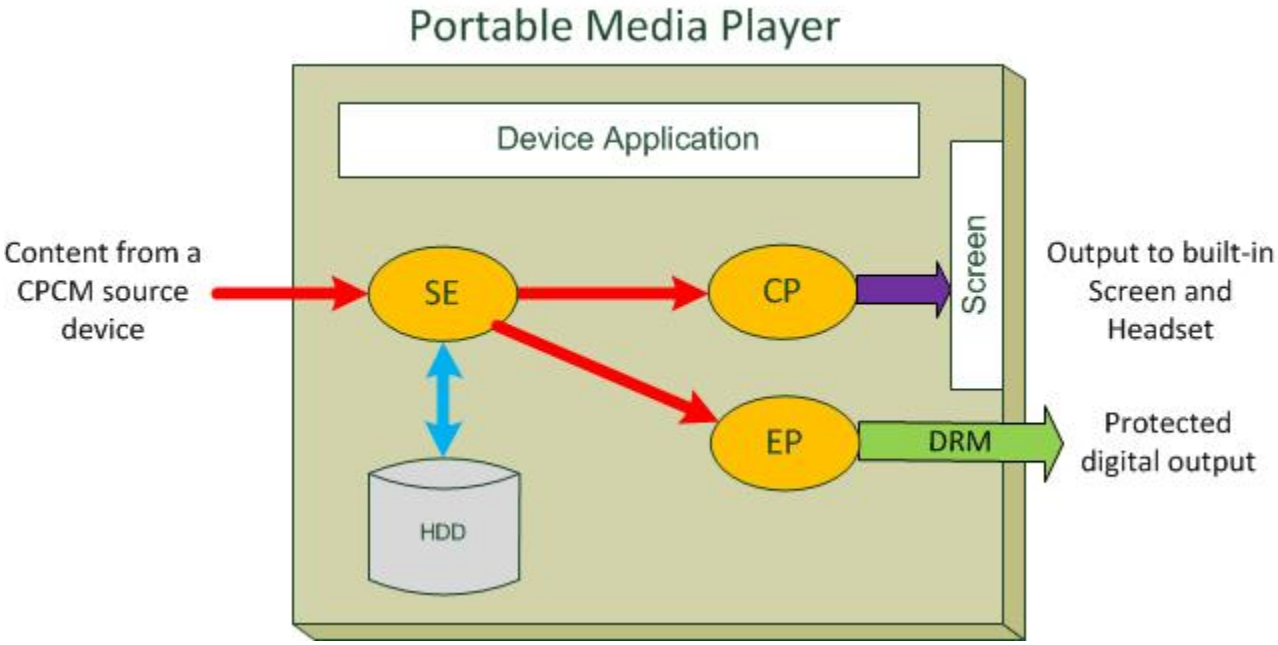


Figure 3: Portable Media Player Example

For this example, the following CPCM Functions are to be implemented:

- Storage Entity, for the PMP local storage
- Consumption Point, for the integrated screen
- Export Point to the CPS used on the digital link

There is also a possibility that the PMP device can provision content for a target device be it CPCM or not; the PMP device supports a resident or downloadable application that can process the content, e.g. down-resolution or transcode. In such case, a Processing Entity Function is needed.

4.2.2 Pay-TV Personal Video Recorder

This example of a PVR is a device that is able to receive content from a Pay-TV operator, to record it locally and to play back the content using an analogue output. The analogue output has an analogue protection system.

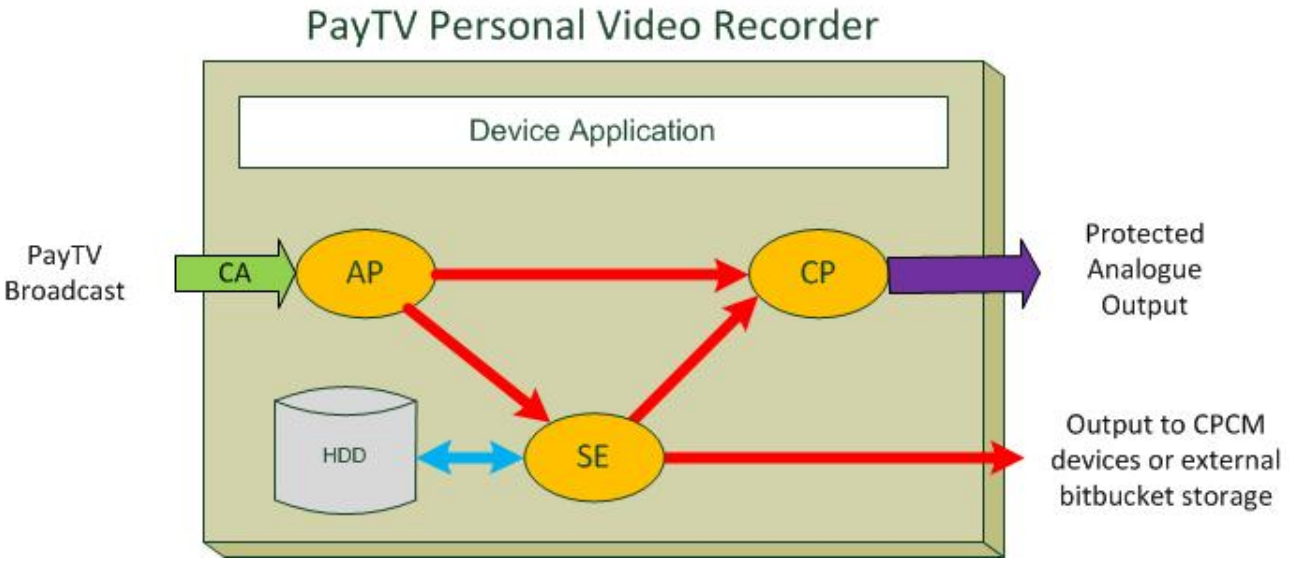


Figure 4: Pay-TV Personal Video Recorder Example

For this example, the following CPCM Functions are to be implemented:

- Acquisition Point, to import content into CPCM from the Pay-TV CAS
- Storage Entity, for the local storage
- Consumption Point, for the analogue output function

Export Point Function is not needed since the analogue output is protected.

4.2.3 Pay-TV Set-Top-Box

In this example, a Set Top Box is a device that is able to receive content from a Pay-TV operator and to play it back using a digital output that includes a protection system.

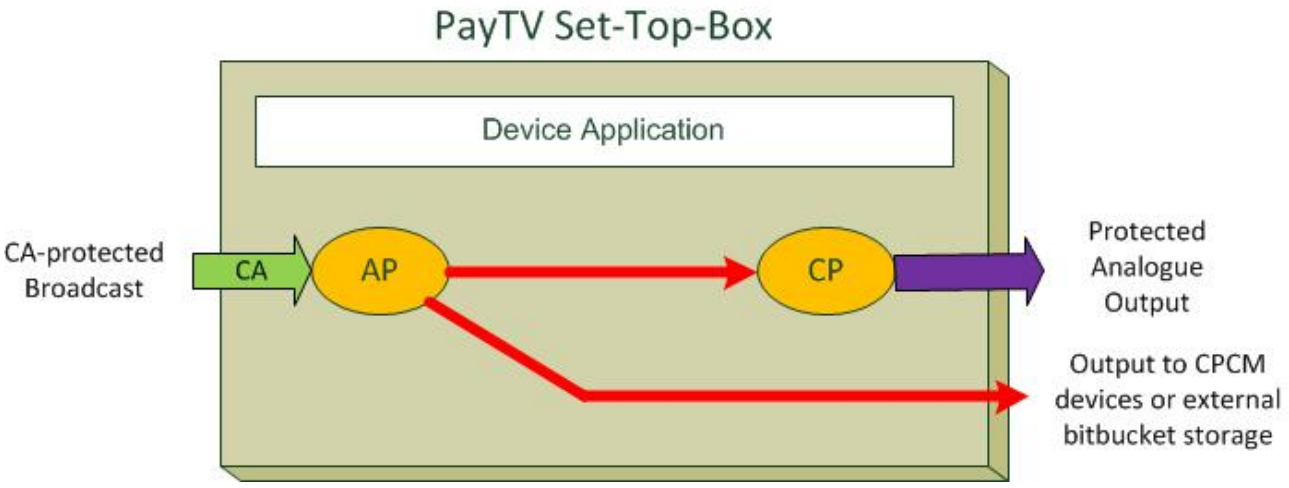


Figure 5: Set-Top-Box Example

For this example, the following CPCM Functions are to be implemented:

- Acquisition Point, to import content into CPCM from the Pay-TV CAS
- Consumption Point, for the digital output function

Export Point Function is not needed since the digital output is protected.

4.2.4 Home Gateway

In this example, the Home Gateway is capable of receiving non-CPCM content, storing it internally or externally and transferring it to a CPCM or a non-CPCM device.

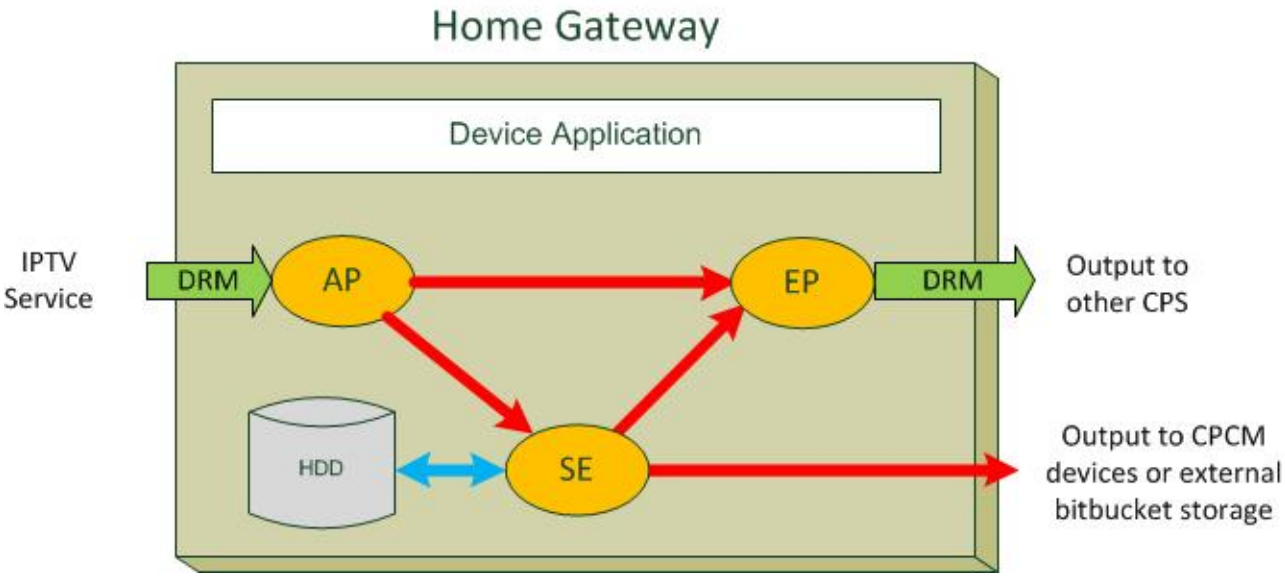


Figure 6: Home Gateway example

For this example, the following CPCM Functions are to be implemented:

- Acquisition Point, to import content into CPCM
- Export Point, to transfer the content to a non-CPCM device
- Storage Entity, for storing the content on internal or networked storage

4.2.5 Home media entertainment Server

This example of a Home Media Entertainment Server is a device that is able to receive content from the air (i.e. it has one or more tuners), from a Conditional Access system (for this example, it is assumed that any Conditional Access functionality is external to the Server), from an Ethernet Interface, a media drive such as CD/DVD/Blu-Ray/HD-DVD, and from a memory card. It has analogue and digital outputs and one or more storage devices. It has the ability to re-format content before storage and/or before passing content to the analogue and digital outputs. High definition Content transmission to a TV screen is protected using HDCP. It may re-transmit content received from the air or from its media drive or memory a card to other devices through its Ethernet interface.

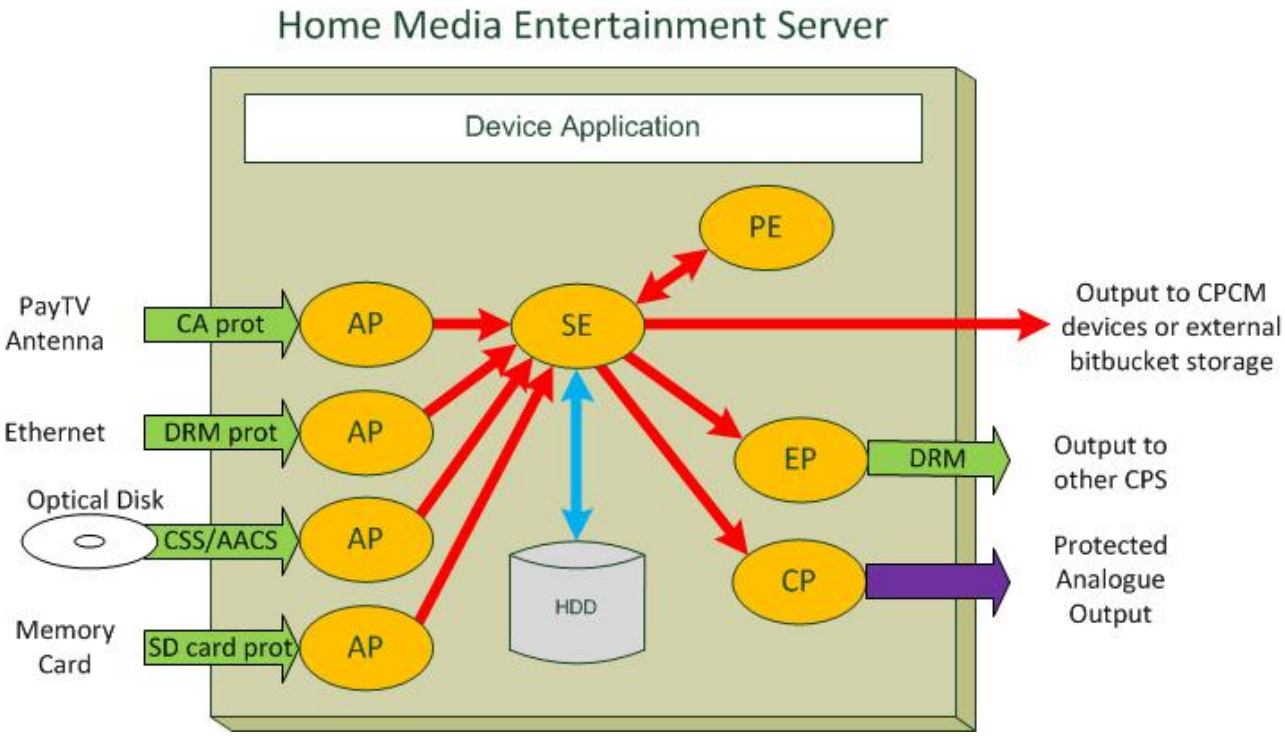


Figure 7: Home Media Entertainment Server Example

For this example, the following CPCM Functions are to be implemented:

- Acquisition Point, to import content into CPCM from the Air, from an Ethernet interface, from a media drive, from a memory card or from a Pay TV CAS.
- Processing Entity, to re-format content before passing content to the Storage Entity(ies).
- Storage Entity, for local storage. Here, CPCM content can be written to an internal storage device such as a hard disc drive and to secure recordable media including DVD, Blu-Ray/HD-DVD or memory card.
- Consumption Point, for analogue and digital output function.
- Export Point for analogue output, unprotected digital output, and export to media drive, memory card and Ethernet interface.

4.2.6 Receiver for unscrambled Audio

This example of Audio receiver is a device that is able to receive digital audio content from a digital input or from the air (i.e. it receives digital audio broadcast). It is only able to receive unscrambled content. It is only able to render the content. It has an analogue output to enable using standard headphones and a digital output to enable using Bluetooth headphones. It has no storage. For some content formats, it may also re-format content so that Bluetooth headphones may be used.

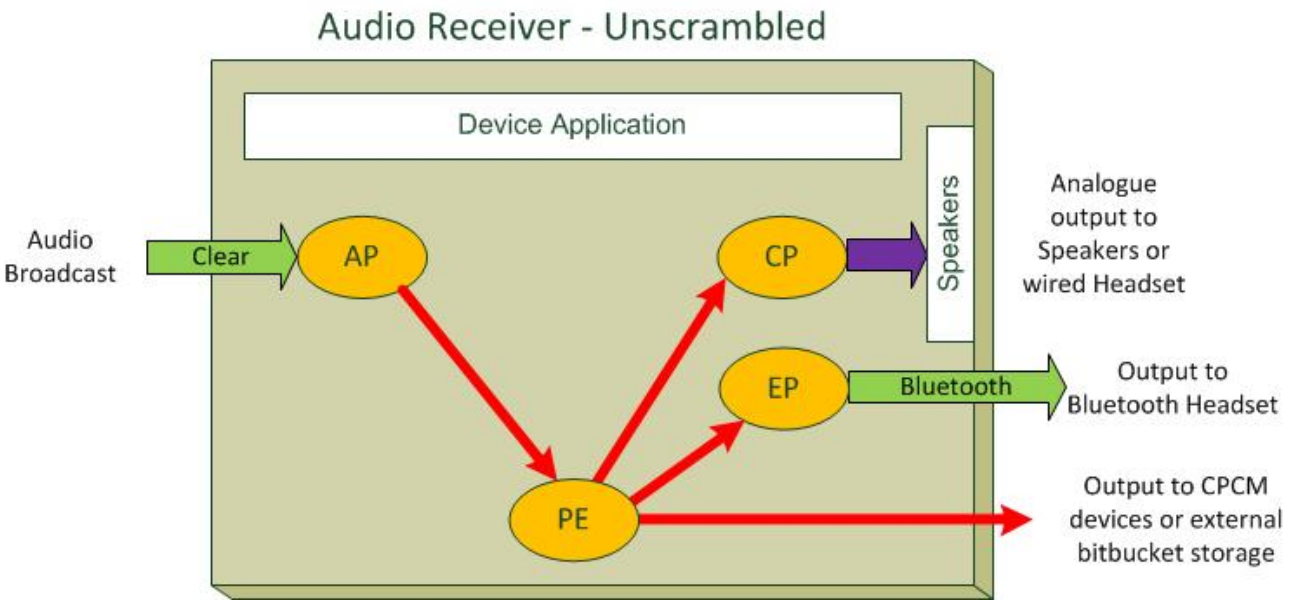


Figure 8: Receiver for Unscrambled Audio Example

For this example, the following CPCM Functions are to be implemented:

- Acquisition Point, to import content into CPCM from the air. The implementation of the Acquisition Point needs only to deal with Free-To-Air content type.
- Processing Entity, to re-format content before passing it to the digital output.
- Consumption Point, to render the content.
- Export Point, to send content to the analogue output (which is not protected) or to the Bluetooth headset.

4.2.7 Receiver for scrambled Audio

This example of Audio receiver is a device that is able to receive digital audio content from a digital input or from the air (i.e. it receives digital audio broadcast). It is able to receive protected or unprotected content. It is only able to render the content. It has an analogue output to enable using standard headphones and a digital output to enable using Bluetooth headphones. It has no storage. For some content formats, it may also re-format content so that Bluetooth headphones may be used.

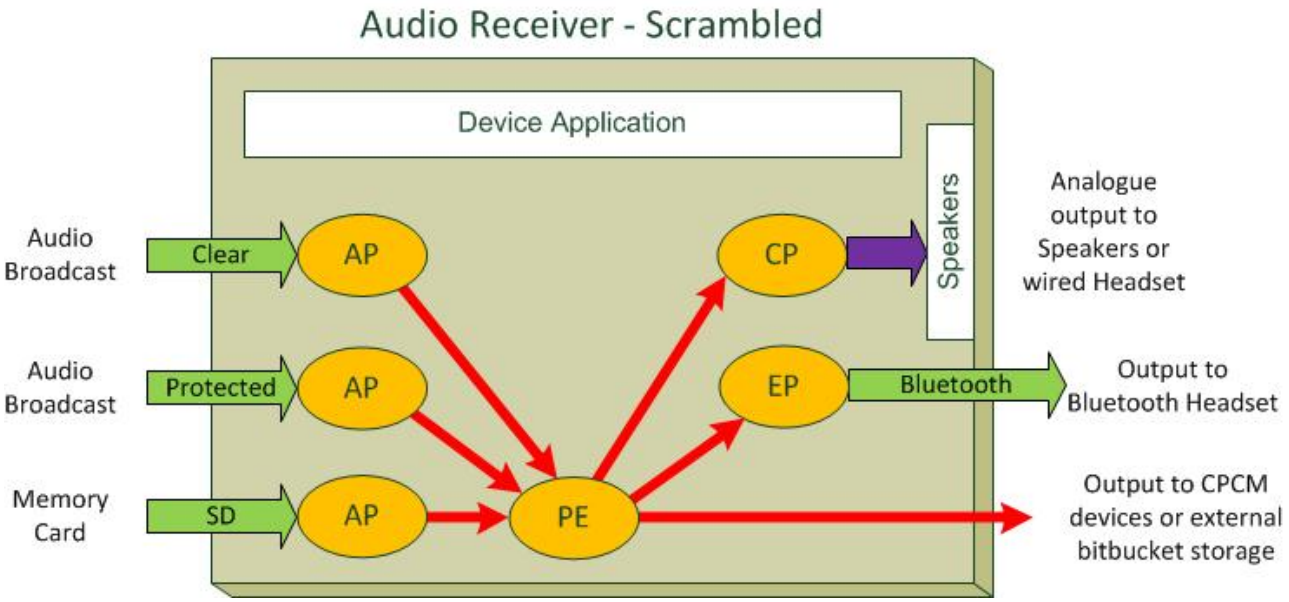


Figure 9: Receiver for scrambled audio example

For this example, the following CPCM Functions are to be implemented:

- Acquisition Point, to import content into CPCM from the air or from memory cards having a built-in content protection system
- Processing Entity, to re-format Content before passing it to the digital output
- Consumption Point, to render the Content
- Export Point, to send the content to the analogue output, which is not protected, or to the Bluetooth headset

4.2.8 CPCM Display Adaptor

This example of a CPCM Display Adaptor is a device that is able to receive only digital video content from a digital input. It has no digital tuner and as such may not receive any digital content from the air. It has no analogue or digital output and no storage. Content transmission to its High Definition output (e.g. HDMI) is protected using HDCP. This allows addressing a non-CPCM compliant HD screen.

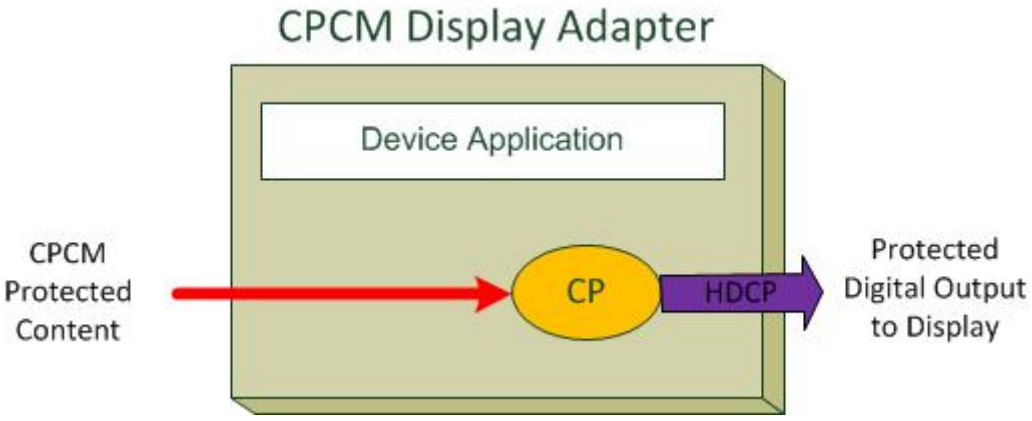


Figure 10: CPCM Display Adaptor

For this example, the only implemented CPCM function is a Consumption Point, which Outputs to a protected digital interface. This is not an Export Point as the output can be used only for viewing purposes.

NOTE: The mapping of FTA signalling towards HDCP is subject to C&R regime as described in TR 102 825-13 [i.5].

4.2.9 Integrated Digital TV Set without network connectivity

This example of Integrated TV-set is a device that is able to receive only digital video content from a digital input. It has a digital tuner to receive Free-To-Air content. It has no analogue or digital output and no storage. Content may only be consumed internally to the device by means of the built-in screen and speakers.

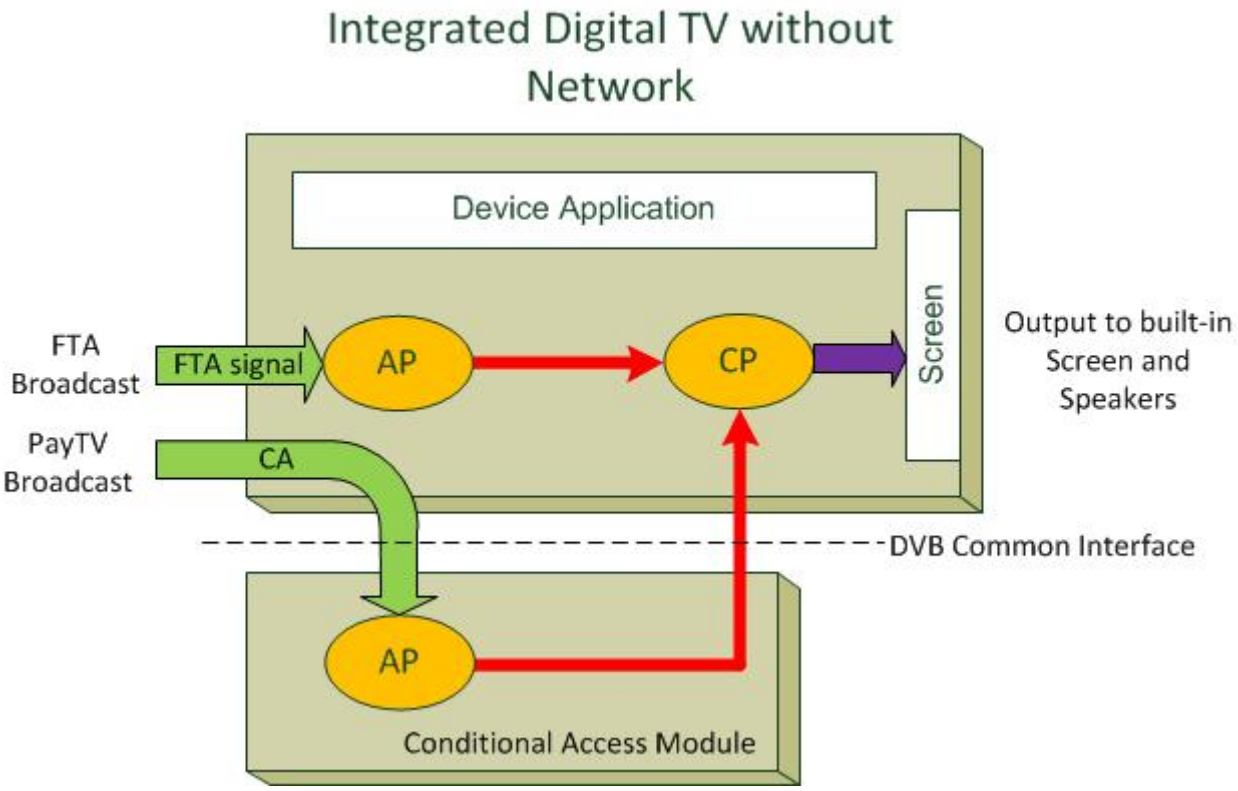


Figure 11: IDTV without network connectivity example

For this example, the following CPCM Functions are to be implemented:

- Acquisition Point, to acquire the Free-To-Air content. Only components relative to Free-To-Air acquisition needs to be implemented.
- Consumption Point, to render the content.

NOTE: CPCM can be used on such a device e.g. to protect CI interface.

4.2.10 Integrated Digital TV Set with network connectivity

This example of an Integrated Digital TV-set is a device that is able to receive content from the air by means of a built-in a tuner. It does not have any analogue or digital audiovisual outputs or storage. It does have an Ethernet interface which may be used to send or receive content to or from other devices.

For this example, the implemented CPCM functionalities are an Acquisition Point and a Consumption Point, which is the screen and speakers. There is no Export Point or Consumption Output. It supports the input or output of CPCM Content over the Ethernet interface.

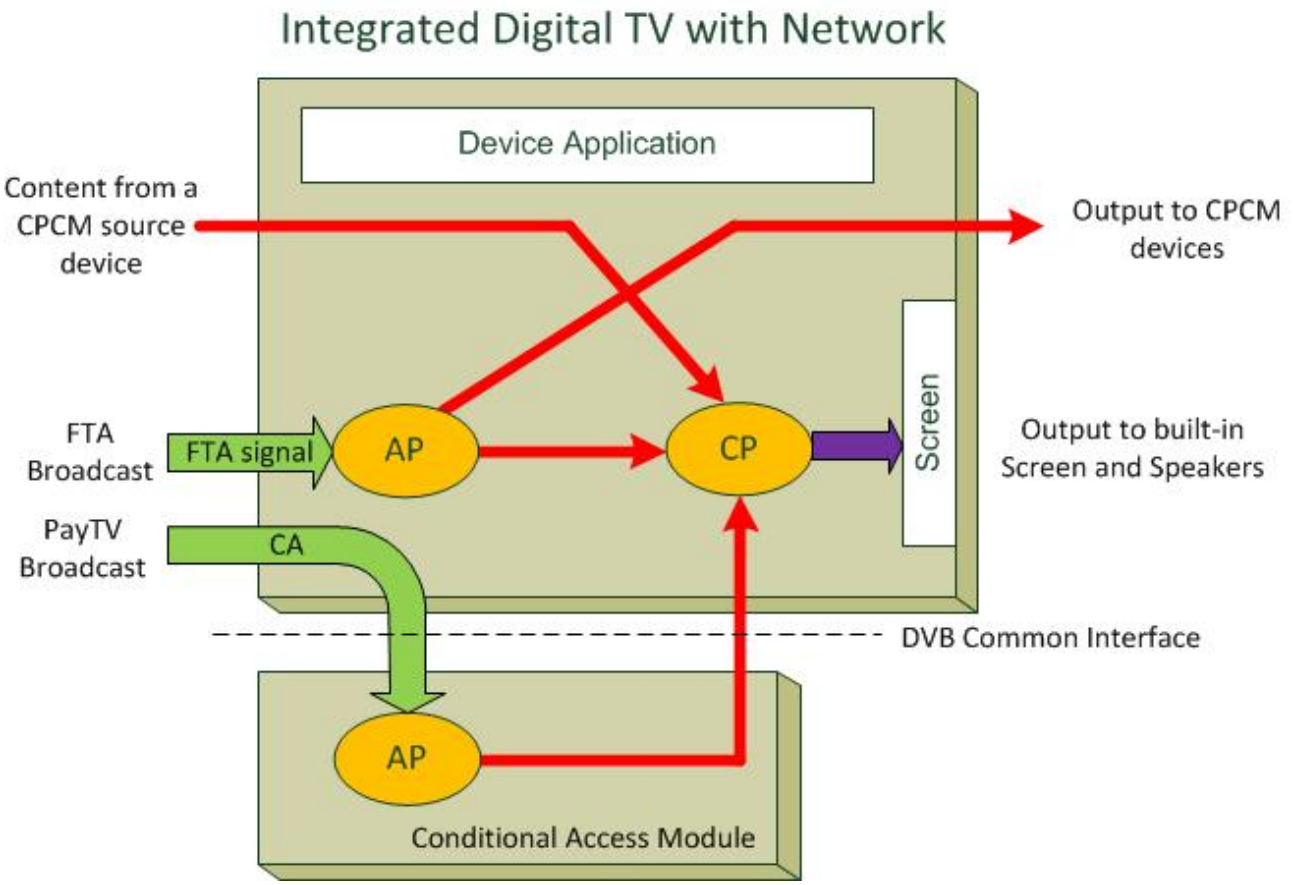


Figure 12: IDTV set with network connectivity

The Ethernet interface may be used to connect to an external Storage Entity in order to store and retrieve CPCM Content. The device may also implement an internal Storage Entity function to enable full-function use of non-CPCM external storage devices as described in clause 4.4.2.

4.2.11 Common Interface Module with Integrated Digital TV including a PVR

In this example, the Common Interface Module is a device plugged in the DVB-CI slot of an Integrated Digital TV (IDTV). The CI Module includes an appropriate Conditional Access for decrypting Pay-TV content. The IDTV includes Personal Video Recorder functionality to record content locally. The IDTV has a network connection for allowing other CPCM compliant devices to access recorded content.

During reception, encrypted Pay TV content is sent from the IDTV to the CI Module for Acquisition into CPCM. CPCM Content is then returned encrypted to the IDTV for Consumption or local Storage with appropriate usage rules as signalled by the USI.

During reception, unencrypted FTA is Acquired by the IDTV before being passed to the CI module. If the Content is marked DNCS (after Acquisition), transmission to the CI module is in the clear; else, the IDTV first CPCM scrambles the Content.

Stored Content is accessible to other CPCM Instances only connected to the IDTV PVR via the network connection.

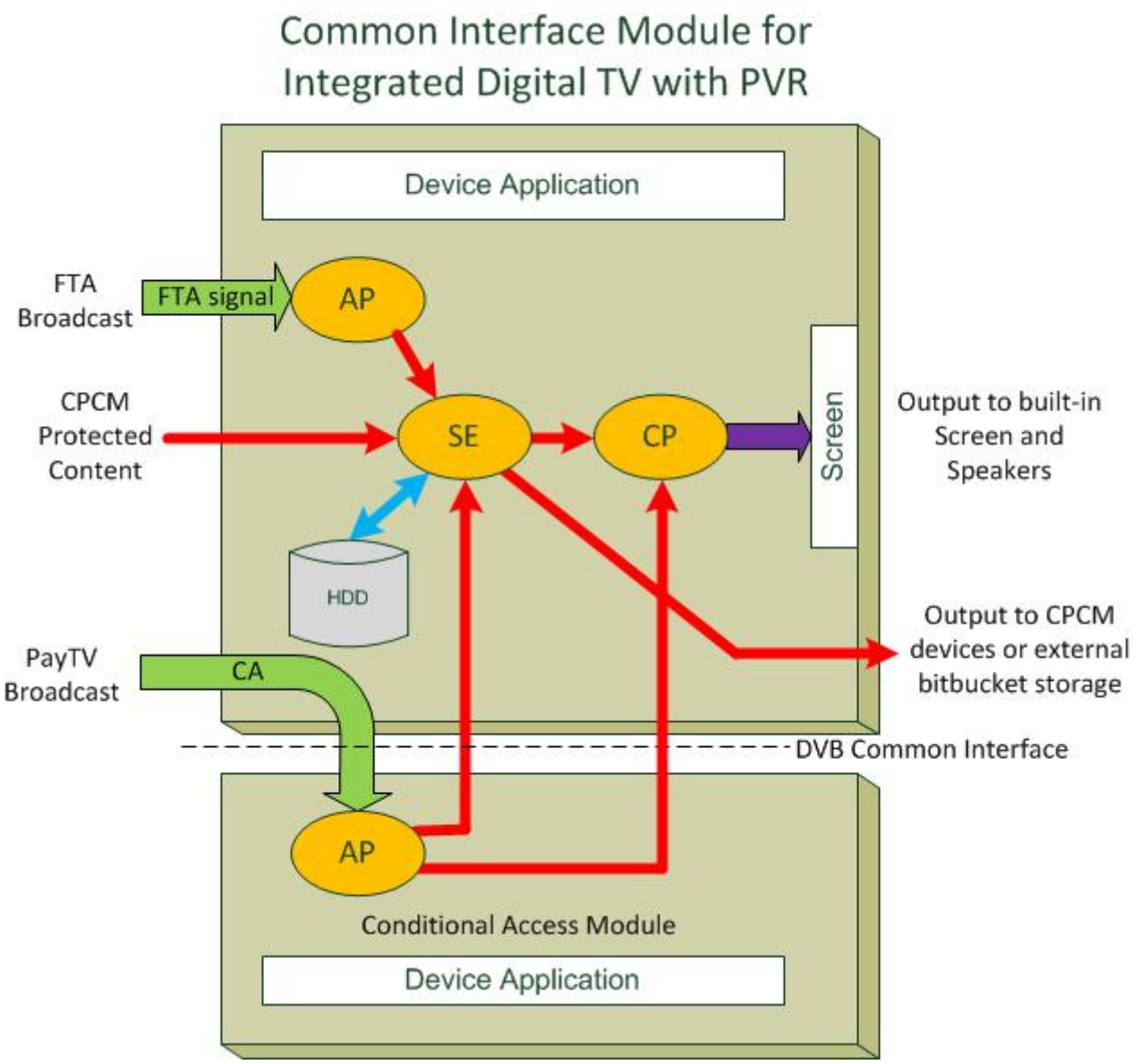


Figure 13: CI adaptor for IDTV with PVR

For this example, the following CPCM Functions are to be implemented in the CI Module:

- Acquisition Point, to import content into CPCM from the Pay-TV CAS or over-air encrypted FTA

For this example, the following CPCM Functions are to be implemented in the IDTV with PVR:

- Acquisition Point, to handle unencrypted FTA content, i.e. clear-to-air
- Consumption Point, for direct content display
- Storage Entity for the local recording

4.2.12 Personal Computer

The Personal Computer needs to be considered as a modular device type, where CPCM functionality may be implemented in the *system* level or in the *application* level, or both. Also, various system components can provide CPCM functionality.

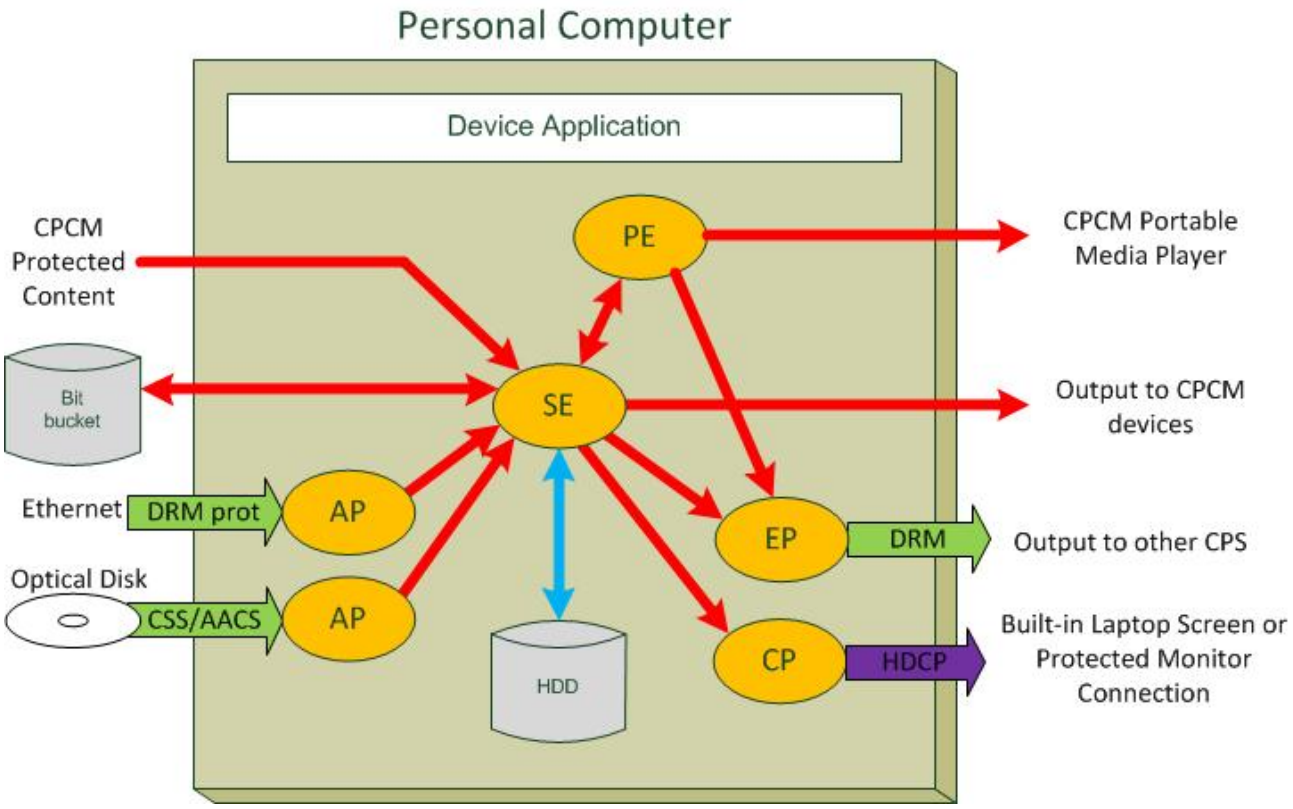


Figure 14: PC Example

Tables 1 and 2 provide examples of how a CPCM Functionality or component can be implemented in a Personal Computer.

Table 1: CPCM Functional Entities implementation in a Personal Computer

CPCM Functional Entity	Implementation in Hardware/Drivers	Implementation in Operating System/Shared Software Utilities	Implementation in Application Software
Notes:	This refers to a physical device and its associated drivers. The O/S might not be aware that this functionality exists.	This refers to implementation of CPCM in the operating system itself, or in common software components that may be reused by many different applications (e.g. shared libraries, managed code environments, and other types of middleware).	This refers to free-standing applications that do not rely on CPCM support from the O/S, middleware, or PC hardware. To provide the necessary robustness, they are likely to make use of generic (non-CPCM) security features of the hardware and operating system
	The O/S may not be aware that an application is CPCM-aware. Multiple CPCM-aware applications may co-exist on the same PC without knowledge of one another. However, each would function as a separate CPCM Instance.		
Acquisition Point	As for other devices, the AP can be implemented in a tuner device, or even in the same chip (e.g. a CA decoder chip)	An AP may be implemented as a system component able to receive clear-to-air content from a basic tuner, or as part of another DRM system (e.g. for Internet content)	An application designed to handle commercial content (e.g. DRM protected or from a clear-to-air tuner) can implement the AP. One example would be a media player for Internet content that can "export" from its native DRM to CPCM.
Consumption Point	It would be possible to build a CPCM CP into a graphics and/or sound card, or in a driver running in a secure environment. This would allow the content to remain under CPCM protection as it flows around the PC.	The Operating System provides management of all graphical and audio output from the PC, so this management could be extended to include CP functionality.	A CPCM-enabled player application can provide CP functionality, then hand the media to the output hardware using existing protected media path technology.
Export Point	Seems unlikely.	Where O/S provides DRM and/or link protection for its output, this function could be expanded to include the CPCM EP function.	An application that provides output of content in protected forms could include the CPCM EP function. Examples could include media players that apply DRM for output streaming (e.g. DTCP-IP, WMDRM-ND), or that protect content for side-loading to portable devices (e.g. WMDRM, OMA DRM, FairPlay).
Storage Entity	Seems unlikely, as PC storage devices are dumb bit-buckets.	An O/S implementation of the SE could employ existing proprietary disk security technology (e.g. partition/file level encryption), or it could leave CPCM encryption in place, or do both as required by C&R.	An application can implement the SE and provide its own management of content stored to the disk, external storage, or removable media.
Processing Entity	Unlikely for consumer equipment.	The O/S could provide some basic PE functions, such as extracting stills and sound levels for usability purposes.	Richer PE functions such as editing, the addition of new media elements (e.g. additional accessibility or language features) are likely to be implemented in authorised software applications.

Table 2: CPCM Component implementation in a Personal Computer

CPCM Component	Implementation in Hardware / Drivers	Implementation in Operating System / Shared Software Utilities	Implementation in Application Software
Authorised Domain Management	Some device types may include separate ADM functionality so they can join the same domain as the PC using CPCM protocols. This is especially true for "pluggable" devices that can be moved between PCs. Others may be more tightly integrated with a single PC and use proprietary means to obtain necessary domain secrets.	The O/S can implement ADM functionality even in the absence of other CPCM functionality.	An application may choose to implement its own ADM functionality, especially if it is intended for use on PCs that do not have CPCM in the O/S.

4.2.13 Mobile Phone

In this example, the mobile phone can store music, including ring tones, in its internal memory and play back using the built-in speaker, analogue headphones or Bluetooth headphones. The music can also be moved to a memory card, where it can still be protected with DVB CPCM or by using built-in content protection technology on the card itself.

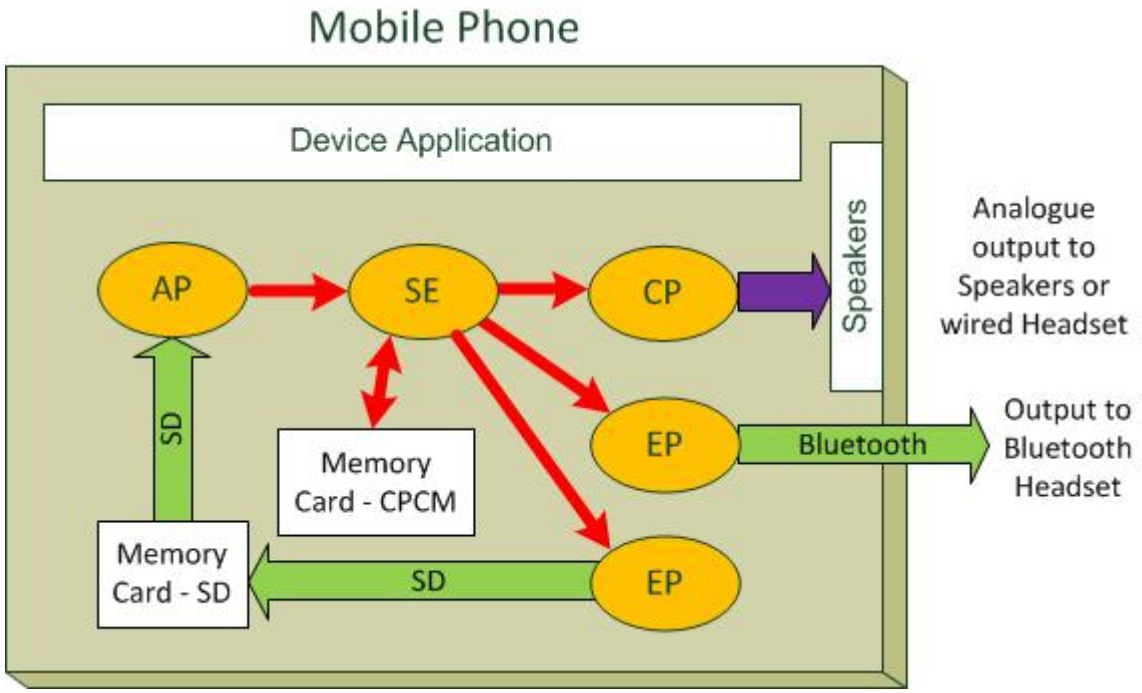


Figure 15: Mobile phone example

For this example, the mobile phone implements the following CPCM Functions:

- An Acquisition Point to read content from memory card having built-in content protection system
- A Storage Entity to deal with internal storage and memory cards
- A Consumption Point, for music play back using the built-in speaker or the analogue headphone
- An Export Point, to deal with built-in content protection technology of the memory card, or to use Bluetooth headset

NOTE: Figure 15 shows two separate SD cards since they have logically different roles. This does not preclude using the same physical SD card.

4.2.14 Mobile TV

In this example, the mobile TV, which may or may not be integrated with a mobile phone, can receive a television broadcast transmitted with IPDC over DVB-H technology. Live TV content can be rendered directly on the small screen on the mobile TV device. The Content can also be recorded on a memory card using DVB CPCM for protection, or it can be exported to a memory card that has its own protection. The live TV playback can also be paused for a short period of time, such as a phone call, and resumed later. In that case, the content is stored in built-in memory. The Content can also be transferred over USB, Bluetooth or WLAN to a PC or a media server (which are other examples of Storage Entities) or streamed directly over WLAN to a nearby renderer, in which case it may undergo some Processing first.

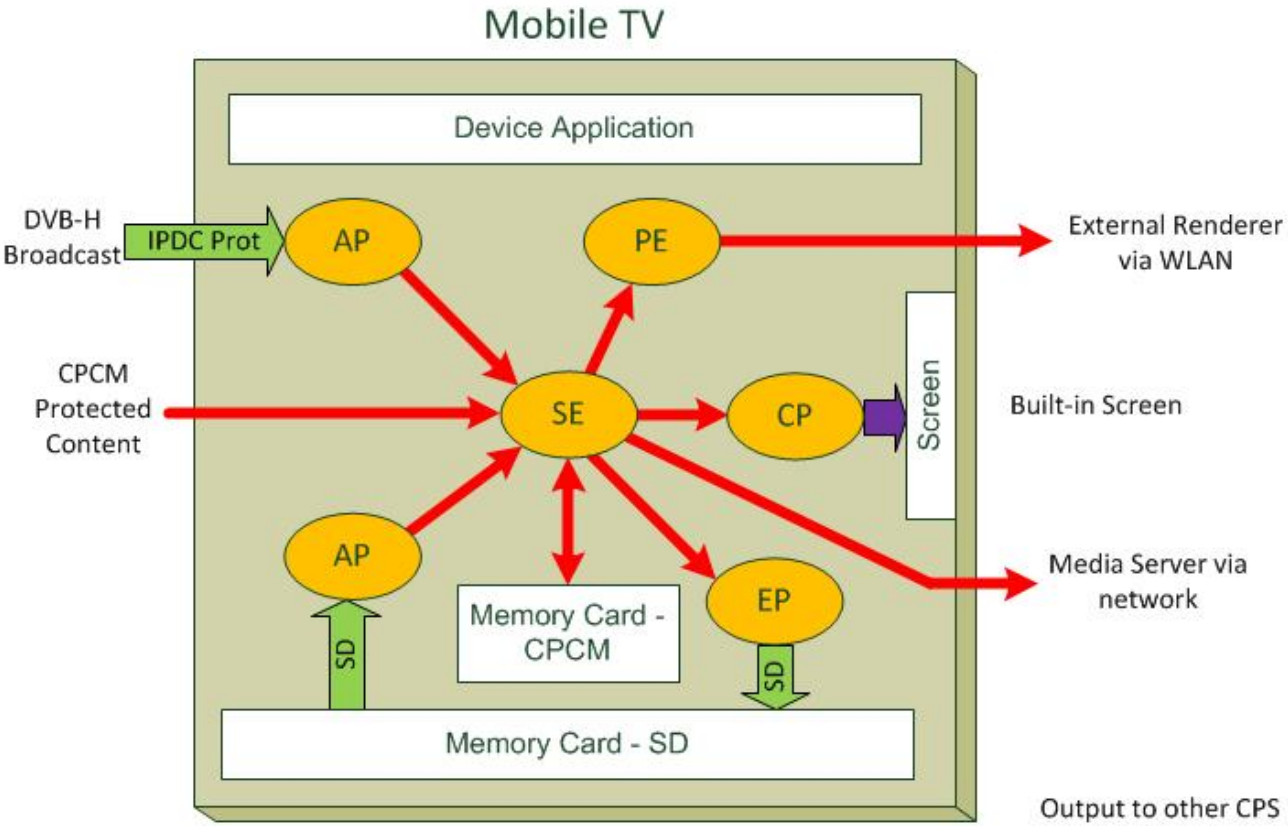


Figure 16: Mobile TV Example

For this example, the mobile TV implements the following CPCM Functions:

- An Acquisition Point to obtain the content from IPDC or from a memory card with built-in content protection system
- A Storage Entity to deal with internal storage, memory cards, media servers…
- A Consumption Point, to play the CPCM Content on the mobile TV screen
- An Export Point, to deal with built-in content protection technology of the memory card
- A Processing Entity to stream the content to a nearby renderer

NOTE: Figure 16 shows two separate SD cards since they have logically different roles. This does not preclude using the same physical SD card.

4.2.15 Common Interface Adaptor for non-CPCM TV

This example of Common Interface Adaptor for non-CPCM TV is a Common Interface Module that is able to receive CPCM content from a digital input and to retransmit it to a legacy TV-set that does not implement CPCM. The CI module provides the necessary home network interface to receive CPCM content.

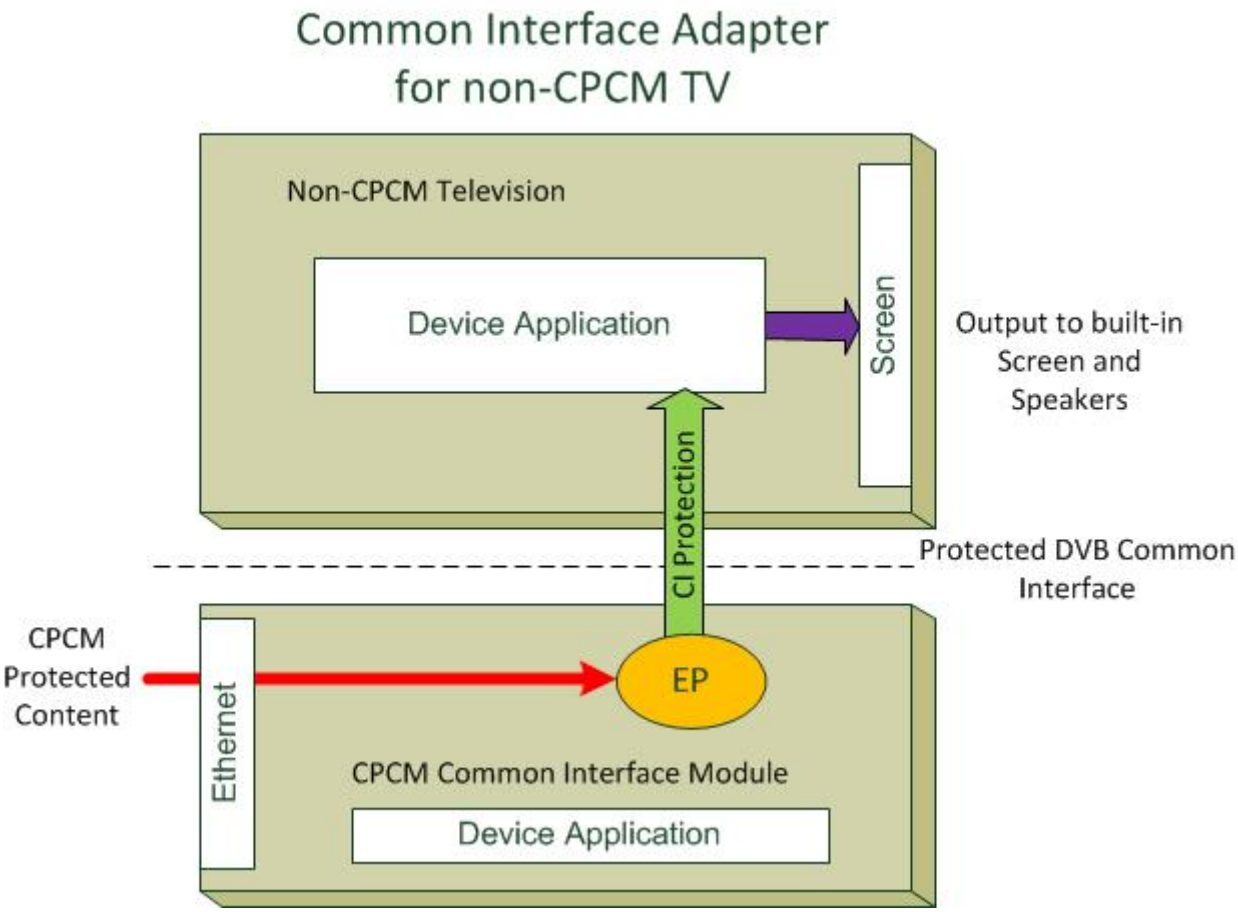


Figure 17: CI adaptor for non-CPCM display example

For this example, the common interface module implements the following CPCM functions:

- An Export Point, to export content towards the common interface.

NOTE 1: Whether the Common Interface is protected or not and with which Content Protection System is a C&R regime choice.

NOTE 2: The TV-set does not implement any CPCM or home network functionality in this example.

4.3 CPCM Content

The notion of CPCM Content Item is different from the generic notion of a content item. A CPCM Content Item is strictly associated with one CPCM Content Licence:

- If for instance crypto-periods are used it may happen that a generic content item includes many CPCM Content Items.
- Similarly, if the same rights and same control words apply to different consecutive events in a streaming channel then they may constitute one single CPCM Content Item.

For scrambled CPCM Content, the association between Content and CPCM Licence is enforced by the presence of the relevant Content Scrambling Key in the CPCM Content Licence.

For clear CPCM Content, the simple presence of a CPCM Content Licence is sufficient to apply the applicable usage rights. The time span and scope of the CPCM Content Licence in such a case is defined by the C&R regime.

CPCM enables revocation on a per CPCM Content basis. The CPCM Content Licence indicates the minimal index of the applicable Content Revocation List (CRL). This ensures that if content was available upon its Acquisition, it will be available even if the applicable CPCM Instance was revoked afterwards. It also allows the enabling of access to less sensitive Content (e.g. adverts) for revoked CPCM Instances.

When accessing CPCM Content, it is recommended that a CPCM Instance refreshes the CRL that is transported with it. This helps in fast propagation of the most recent CRL. This will never negatively affect the associated Content but will ensure an up-to-date transmission of Content Revocation Lists. In some cases, it may even provide more possibilities to access the Content (e.g. if the CPCM Instance identifier is no more in the CRL).

4.4 CPCM Content Management

Table 3 details the CPCM elements which are to be implemented with regard to the CPCM functions implemented in the device. This is not bound to any hardware architecture.

Table 3: CPCM element implementation vs. CPCM functionalities

CPCM element	Acquisition Point (DNCS only)	Acquisition Point (general case)	Processing Entity	Storage Entity	Consumption Point (DNCS only)	Consumption Point (general case)	Export Point
SAC	C&R option	Yes	Yes	Yes	C&R option	Yes	Yes
Certificate verification	C&R option	Yes	Yes	Yes	C&R option	Yes	Yes
Revocation management	C&R option	Yes	Yes	Yes	C&R option	Yes	Yes
CPCM scrambler	Optional	Yes	Yes	Optional	No	No	No
CPCM descrambler	No	No	Yes	Optional	No	Yes	Yes
Content decoder	No	No	No	No	Yes	Yes	Output dependant
Proximity Tools	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ADM	Recommended	Recommended	Optional	Optional	Optional	Optional	Optional
Content Licence issuing	Yes	Yes	Yes	Yes	No	No	No
Content Licence verification	No	Optional	Yes	Yes	Yes	Yes	Yes
Secure Relative Time	If SAC is implemented	Yes	Yes	Yes	If SAC is implemented	Yes	Yes
Secure Absolute Time	Optional	Optional	Optional	Optional	Optional	Optional	Optional
Geographical awareness	Optional	Optional	Optional	Optional	Optional	Optional	Optional

Besides elements that are always optional, the following may help an implementer to determine whether optional elements are required:

- Requiring SAC for DNCS only devices is a C&R option: A C&R regime will require DNCS only devices to implement SAC if it requires strong authentication before exchanging DNCS content.
- Requiring certificate verification for DNCS only devices is not an option if a SAC is required. Else, a C&R regime will require DNCS only devices to implement certificate verification if it requires weak authentication before exchanging DNCS content.
- Revocation management is useful in CTA receivers only in markets where content might be signalled with do_not_apply_revocation unset.

- CPCM scrambler is useful in CTA Acquisition Point only in markets where content might be signalled with do_note_scramble unset.
- Storage Entities do not need usually to implement CPCM scrambler or descrambler. However, if they want to be able to re-apply one unique Content Scrambling key to the whole content (maybe needed e.g. in the follow-me scenario), they need to implement both.
- Content Licence verification may be needed in Acquisition Points (or in AAA) able to do Content Licence Re-acquisition.
- Secure Relative Time is needed as soon as SAC is implemented.
- ADM implementation is optional but recommended for Acquisition Points as the AD information should be instantiated in the Content Licence at that stage.

A CPCM Instance is not supposed to alter or remove any non-CPCM information, such as SRM or other CPS metadata that would be transported together with the content.

4.4.1 Acquisition

Elements necessary to implement an Acquisition Point are shown in table 3.

An Acquisition Point (dealing with DNCS signalled and other content) is not required to implement all four modes of the CPCM scrambler. If the Trusted Sources from which the Content is Acquired only uses a subset of the modes, then the Acquisition Point may only implement these modes. An Acquisition Point only dealing with DNCS content is not required to implement a CPCM scrambler.

The interface between the Acquisition Point and the content Trusted Source is not defined by CPCM. Acquisition Point implementers may define their own interface subject to agreement between the C&R regime and the regime for the other for both technologies.

An Acquisition Point obtains rights information from Free-To-Air signalling or from the CPS from which it Acquires the content. When content is Acquired from Free-To-Air it is recommended that the Acquisition Point is recommended to preserve the original Free-To-Air signalling and includes the signalling in the relevant section of CPCM auxiliary data. The original signalling may be used for user information while the signalling included in the CPCM auxiliary data may be used when Content is transferred to another Content Protection System upon Consumption or Export.

When content is Acquired from an external CPS, the Acquisition Point may preserve the original Content Licence. This will help maintain unambiguous transfer of original rights back to the same CPS or to another CPS. Additionally, the Acquisition Point may insert in the auxiliary data any information proprietary to the source CPS or relevant to any CPS to which the content would be later transferred.

4.4.2 Storage

There are several ways to deal with storage using CPCM:

- Implementing a Storage Entity Function and storing the content internally to the device. CPCM Content is handled by the Storage Entity Function upon storage and retrieval. The Content Licence can be stored using any protection mode allowed by the USI setting.
- Implementing a Storage Entity Function and storing the content externally to the device (including removable storages). CPCM Content is handled by the Storage Entity Function upon storage and retrieval. The Content Licence can be stored using any protection mode allowed by the USI setting. In the case that the Content Licence was recorded under ADS control only, the Storage Entity that retrieves the content is not necessarily the same as the one that recorded it (bit bucket scenario).
- Not implementing any Storage Entity Function in which case only the bit bucket scenario is supported. The Content Licence has to be only protected by the ADS. The Content and Content Licence are recorded and retrieved without any modifications.

Elements necessary to implement a Storage Entity are shown in table 3.

4.4.3 Consumption

Elements necessary to implement a Consumption Point are shown in table 3.

A Consumption Point has to implement all four modes of the CPCM descrambler in order to be able to deal with all potential Content sources.

4.4.4 Processing

Elements necessary to implement a Processing Entity are shown in table 3.

A Processing Entity needs to implement all four modes of the CPCM descrambler in order to be able to deal with all potential Content sources. It needs also to implement all four modes of the CPCM Scrambler as it cannot change the scrambling mode unless authorized by the C&R regime.

4.4.5 Export

Elements necessary to implement an Export Point are shown in table 3.

An Export Point has to implement all four modes of the CPCM Descrambler in order to be able to deal with all potential content sources.

The interface between the Export Point and the system to which it is exporting is not defined by CPCM. Export Point implementers may define their own interface subject to C&R regimes for both technologies.

4.5 CPCM Content Licence Management

CPCM Content Licences can be transmitted within a CPCM Content Item which is in-band transmission, or in a dedicated message which is known as out-of-band transmission. Auxiliary Data are always transported using the same transmission mode as the Content Licence. Auxiliary Data and the Content Licence are inserted in the same CPCM message that is itself transmitted in-band or out-of-band.

The decision on the transmission mode is implementation dependant. It will probably be chosen by the Source CPCM Instance when Content is pushed onto another CPCM device and by the Sink CPCM Instance when Content is pulled from another CPCM Instance.

CPCM Instances are required by TS 102 825-2 [i.4] to support both modes, but both modes are not required to be supported for all CPCM Content Items, as it may be inefficient for implementations and some content management scenarios may not support both modes.

EXAMPLE 1: If a CPCM Instance received MPEG2-TS content with out-of-band Content Licence, providing the same content with in-band Content Licence would require the implementation to re-multiplex the entire MPEG2-TS.

In push mode, the source CPCM Instance can make the selection regardless of the Sink CPCM capability since the latter supports both modes.

In pull mode, two scenarios may occur:

- The HN layer signals which CL transport modes are supported for the Content Item. In this case the Sink Instance may choose either of the available modes.
- The HN layer does not provide this signal in which case the Sink Instance may try either method, and if it fails, fallback to the alternative mode.

NOTE 1: It may improve the user experience to buffer a received content stream while the Content Licence is obtained.

EXAMPLE 2: A sink CPCM Instance may first ask for Content Transmission, look whether the Content Licence is in-band, and, in the contrary case, request for an out-of-band Content Licence.

When both alternatives are available, the following guidelines may be taken into account while selecting the mode:

- In-band mode is more appropriate in the following cases:
  - MPEG2-TS streamed content, e.g. live television streams
  - Multicasting of Content whose Content Licence is AD protected
  - Content with short crypto-periods, to avoid frequent out-of-band licence requests
  - Content Storage on a bit-bucket, to avoid accidental separation of the content from the Licence, should the file be copied or moved
- Out-of-band mode is more appropriate in the following cases:
  - DVB-FF content
  - Multicasting of Content whose Content Licence is SAC protected:
    - as there will often be room for only one Content Licence in-band of the content
    - as each receiver uses a different SAC key, the multicast would need to include at least one Content Licence per receiver, the number of which may change over time
    - as all multicast Content Licences will have to be inspected by all recipients (there is no means to know which Content Licence is protected by which SAC)
  - Copy-Once content

NOTE 2: It is possible to transfer CPCM protected content using multicast techniques even when the USI settings require SAC protection of the CL. However, this means that the CL needs to be delivered out of band and not embedded in the content stream.

4.6 CPCM and private Extensions

4.6.1 Extension Definition

CPCM supports two kinds of extensions:

- CPCM Extensions that are standardised extensions and published in TS 102 825-14 [i.6]. These CPCM extensions are allocated an identifier (CPCM_extension_identifier).
- Proprietary Extensions that may be defined and implemented by any CPCM implementer. Proprietary Extensions are also allocated an identifier (private_data_identifier).

Developers of CPCM and proprietary extensions may use extensions in two different ways:

- They may define new messages. In this case, they use CPCM_extension_message or private_data_message. The first element carried in the message is the allocated identifier. The following element is extension specific data whose format is specified by the extension developer. If the extension developer needs to specify several messages, it will specify an extension format that allows for this differentiation.
- They may define extension elements to be added to existing CPCM messages. Any CPCM message can carry one or more optional elements. The first element of this optional element is the allocated identifier. The following element is extension specific data whose format is specified by the extension developer. If the extension developer needs to specify several optional elements, it will specify an extension format that allows for this differentiation.

4.6.2 Extension trust

There are two means for a CPCM or a proprietary Extension to be trusted:

- The extension may have its own CPCM Certificate. In this case, CPCM Extension is considered to be a distinct CPCM Instance and trust is managed as for any Extension.
- The trust may be provided by inherent compliance. This is the case, for instance, if the Extension implementation is in the same security environment as the CPCM Instance implementation it is extending. In this case, this CPCM Instance and the Extension are viewed as a single CPCM Instance and use the same certificate. A C&R regime will decide what constitutes the same security environment and how inherent compliance may be achieved.

EXAMPLE: A security environment may be defined by a single hardware element.

4.7 CPCM Technical Compliance & CPCM Interoperability

4.7.1 CPCM Technical Compliance

This multi-part deliverable does not define which CPCM specification elements are mandatory in an implementation and which are optional. Implementations, which include the specification elements that are needed to achieve their expected functionality, will conform to the relevant specification. A C&R regime may in addition require the implementation of additional elements from CPCM implementers.

CPCM can be used as a toolbox. A C&R regime may choose to use CPCM elements in combination with non-CPCM elements.

4.7.2 CPCM as an Interoperability Solution

CPCM is designed to provide a common point of interoperability between conformant devices, and between content protection mechanisms.

The purpose of the CPCM specification is to provide means for protecting both Free-To-Air and Pay-TV content within the home.

As such, CPCM works on a basis of Acquiring content from one or more trusted sources, exchanging the content between open standard and/or proprietary devices, and controlling how the content is then consumed or exported to other systems.

4.7.2.1 CPCM as a Device Interoperability solution

CPCM is intended to enable authorised content sharing between conformant devices. To enable this feature, CPCM devices will implement the following tools for each CPCM Instance:

- Certificate and Revocation status verification
- SAC establishment
- CPCM scrambler and descrambler, where applicable (see table 3)
- Content Licence generation and verification, where applicable (see table 3)
- USI enforcement

4.7.2.2 CPCM interoperability with other CPS

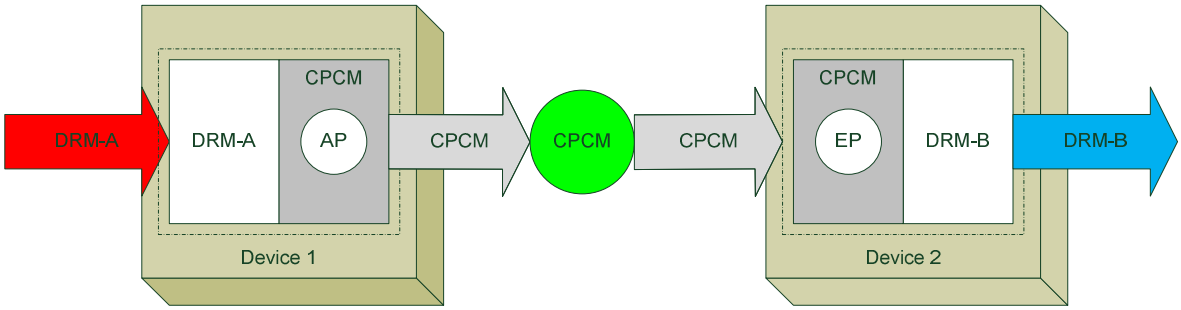


Figure 18: CPCM interoperability with other CPS

CPCM supports Acquisition of content from any CPS, subject to agreement between the relevant C&R Regimes. Similarly, if authorised by the relevant C&R regimes, CPCM supports Export to another CPS.

The mappings between CPCM and the external CPS are also defined under bilateral agreements between the relevant CPCM C&R regime and the external CPS C&R regime.

Mapping at the Acquisition Point is applied by the source CPS from which the content is Acquired. The CPCM Acquisition Point does not need to be aware of this mapping. The security of the implementation of the interface between the source CPS and the CPCM Acquisition Point is subject to agreement between the two C&R Regimes.

Mapping at the Export Point is applied by CPCM. Therefore, the CPCM Export Point applies and enforces the mapping defined by the relevant C&R regimes for each of the external CPS to which it is able to Export.

4.7.2.3 CPCM bridging between CPS

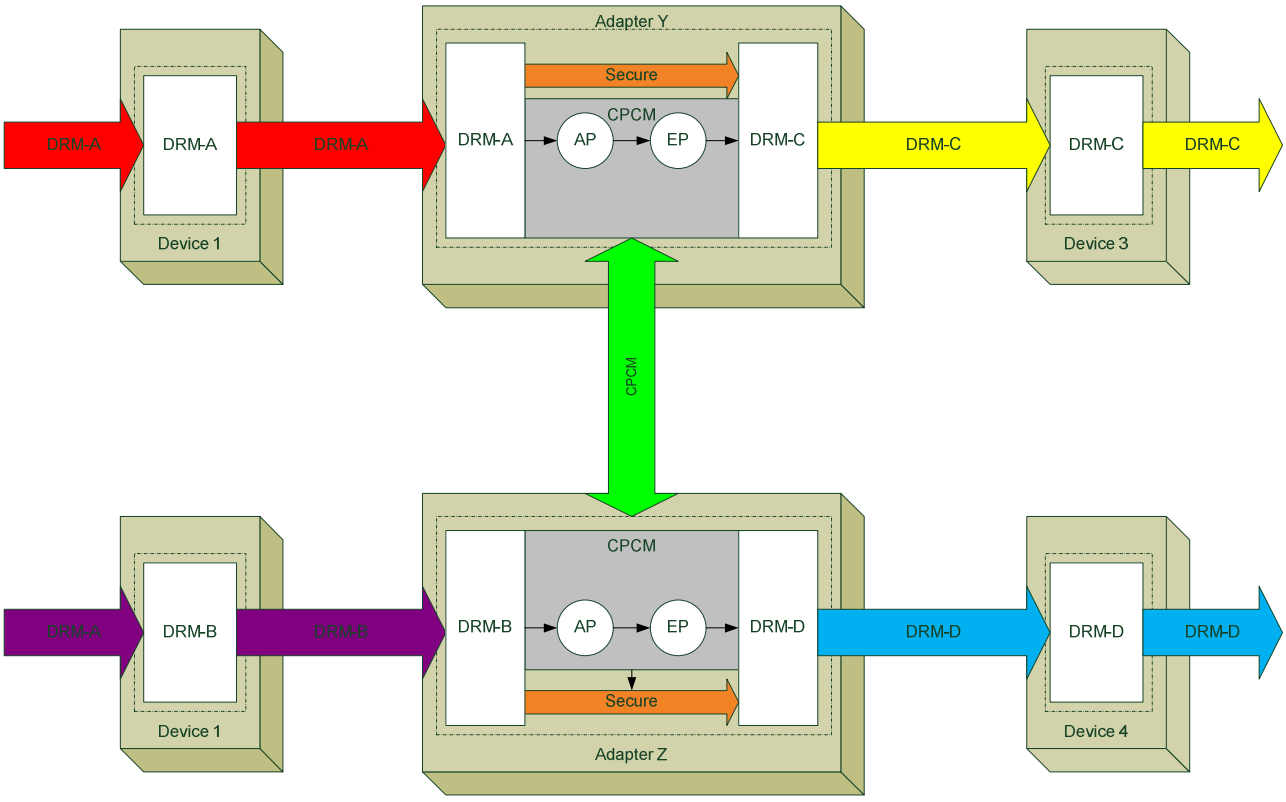


Figure 19: CPCM as a bridging solution

CPCM supports acquisition of content from any trusted CPS, subject to acceptance of the source CPS to deliver content to CPCM. Similarly, if authorised by the relevant C&R Regime, CPCM supports export of content to another CPS. Thus, CPCM may be used as a trusted bridge between two CPSs which are able to acquire and/or export content from/to CPCM.

Using CPCM as a bridge avoids the need to define mappings and trust establishment between each of the different possible pairings of CPS. Each CPS only needs to specify its mappings to the CPCM Acquisition Point and import mappings from the CPCM Export Point in order to be able to interoperate with other CPSs which have similarly defined mappings to/from CPCM.

When content is Acquired into CPCM, it is possible for the Acquisition Point to include the original rights information, or any other information, in the CPCM auxiliary data. While this will not be used by CPCM itself, this information will remain attached to the content and will be available to external CPSs at the time of export from CPCM. This allows a CPS with richer capabilities to take advantage of more complex rights structures than CPCM can handle.

To use CPCM as a CPS interoperability solution, implementers will implement the following tools:

- Certificate and Revocation status verification
- SAC establishment
- CPCM scrambler and descrambler, where applicable (see table 3)
- Content Licence generation and verification, where applicable (see table 3)
- CPCM Acquisition from the source CPS
- CPCM Export Point to the destination CPS

USI support is required but the range of USI may be restricted to a very small subset of the full USI if the rights pass through mechanism is used. For instance, the source CPS might configure the USI as follows:

- CCI is set to 'Copy Never Zero Retention Asserted' to ensure that no copy is retained within CPCM
- Consumption Control is set with 'Viewable' non asserted to ensure that content may not be used within CPCM
- Propagation Control is set to 'MAD' to be sure interoperability is limited to CPCM Instances within a same domain
- Export Flags are all unasserted. The mapping carried in CPCM Auxiliary Data will be the basis for the export.

CPCM Acquisition Points can be implemented to always configure the USI as above and Export Points to only accept content whose USI has the above settings. In this way, the need to implement the full range of USI is minimised but CPCM can still function as an interoperability solution. If the full range of USI is not implemented during the acquisition and export mappings from/to the CPSs, then interoperability between the CPSs will be dependent on the mapping defined between the pairing of CPSs. To avoid this, implementers of CPCM wishing to achieve CPS interoperability are advised to carefully choose the range of USI they want to support and enforce.

5 Usage State Information

5.1 Introduction

This clause is intended to guide the content provider in the expression of content usage using CPCM USI and to further aid the implementer in setting the relevant controls for a given USI configuration. Standard examples of USI settings are given in TS 102 825-3 [i.7] and TR 102 825-11 [i.8]. TR 102 825-11 [i.8] also details how content will be handled in a CPCM environment with respect to various content management scenarios and the corresponding USI settings that are expected for these scenarios.

This clause focuses on four additional aspects of USI configuration namely:

- Equivalent settings: This clause identifies different USI settings that will induce the same behaviour. In such cases, the implementer may unify the treatment of these different settings and the content provider may choose to use the setting corresponding to the usage expectation.
- Unusual settings: This clause identifies combinations of settings that are not expected to be used and show what would be the expected behaviour if they should for some reason be used.

- Bit Bucket Scenario Support; this clause identifies settings that can be used to enable bit bucket usage.
- Corrupted USI: TS 102 825-3 [i.7] gives a normative statement on what needs to be done when some USI are discovered to be corrupted. This clause explains how to apply this normative statement.

5.2 Common misunderstandings

5.2.1 The necessity of thoughtfully setting Propagation Information

When not interested by facilities offered by the AD, one might be tempted to only apply MLocal and VLocal USI if one only wishes to apply proximity control. However, the propagation information needs to be set and it will be acted upon. If the propagation information is set to MCPCM or VCPCM, proximity control becomes irrelevant (see clause 5.3.1) for the purpose of respectfully moving and viewing the content.

If the intended usage is to enable the moving of Content only to Local CPCM Instance, the relevant setting is MLAD and MLocal. This setting does not require CPCM Instances to be members of an AD in order to access the Content but requires setting a Remote Access Rule to lock the Content to the Local Environment for the expected amount of time.

5.3 Equivalent settings

5.3.1 MCPCM and MLocal or VCPCM and VLocal

Content marked MCPCM may be Moved or Copied to any CPCM Instance, regardless of whether it is Local or not. Thus, asserting MLocal does not change permitted usages of Content. In the interest of having consistent USI settings for common behaviours, the Content provider is recommended to always assert MLocal when MCPCM is asserted.

The above applies for the same reasons to VCPCM and VLocal.

5.3.2 Viewing Propagation Information more restrictive than Moving Propagation Information

If Content is marked CCNA and Viewable, then setting Viewing Propagation Information to a more restrictive state than Moving Propagation Information results in the same behaviour as if Viewing Propagation Information was the same as Moving Propagation Information. The Content can actually be Moved or Copied with respect to the Moving Propagation Information settings, and can then be Viewed immediately.

EXAMPLE 1: Content marked VLAD and MAD has the same effective usage restriction as content marked VAD and MAD but encumbers the implementation to move first then view.

A more useful combination is to set Moving Propagation Information to a more restrictive state than Viewing Propagation Information.

EXAMPLE 2: If Content is marked VAD and MLAD, content cannot be Moved or Copied outside of the Local AD until proximity restriction expires but can be viewed remotely without retaining the copy.

The same considerations as above apply for Content that is marked MLocal but not VLocal.

NOTE: If content is marked Copy Once or Copy No More, then setting the Viewing Propagation Information to a more restrictive set than the Moving Propagation Information is more useful. One could use it to ensure that the single copy that is able to be moved around within the Authorized Domain is only viewable within one Local Environment.

5.3.3 Copy Never and Moving Propagation Information

If Content is marked Copy Never, Moving Propagation Information is not relevant, as Content cannot be stored. In such a case, it is recommended to set Moving Propagation Information to the same restriction as Viewing Propagation Information (see clause 5.3).

EXAMPLE: If Content is Marked Copy Never and VGAD, it is recommended to set Moving Propagation Information to MGAD for the sake of consistency, symmetry and overall elegance.

NOTE: If the Content is marked Copy No More, Moving Propagation Information restricts the area where the content can be Moved and is thus relevant.

5.3.4 Time-Restricted content

Content may be simultaneously marked VWA and VPA. In such a case, regarding the time of first consumption, the expiration time will be the most restrictive one as described in TS 102 825-4 [i.9].

Consequently, setting the View Period to a time longer than the length of the Time Window results in the same restrictions as not setting the View Period at all.

EXAMPLE: If the Time Window is set to May and the View Period from first Playback is set to two months, and the first playback of content is on May 5, the expiry will still be effective on May 31.

To avoid confusion, VPA and VWA should only be both asserted in cases where the View Period is shorter than the length of the Time Window.

Moreover, asserting VPA would restrict the user's ability to copy the Content because the act of copying the Content would necessarily start the View Period even though it is not actually Viewed. In such a situation, View Window usage is likely to be preferred to View Period usage as it provides the user with a better experience, since the view window is not triggered by copying and thus requires less user interaction.

5.3.5 Geographical Information

Geographical Information in Auxiliary Data will only be enforced if the content is marked MGAD or VGAD. Consequently, it is generally unnecessary to insert such information if the Content is not marked MGAD or VGAD.

5.3.6 Remote Access Rules

If remote_access_date_immediate_flag and remote_access_date_moving_window flag are both asserted, the less restrictive rule applies, and the behaviour is the same as if only remote_access_date_immediate_flag was asserted.

EXAMPLE: If the content lasts 2 hours and is Acquired at noon, and is available for remote access at 6 pm with both remote_access_date_immediate_flag and remote_access_date_moving_window set, the entire content will be available remotely at 6 pm, the same behaviour as if only remote_access_date_immediate_flag had been set. If only remote_access_date_moving_window was set, the content in its entirety would not be available until 8 pm.

NOTE: The above does not apply for the CPCM delivery signalling but the Acquisition Point is expected to assess which rule first applies and to set only the corresponding date (see clause 10.3).

5.3.7 Time-restricted content which also has proximity or geographical restrictions

If content is time-restricted and also geographically or proximity restricted, then a Remote Access Rule that enables Remote Access after the Content Licence has expired will never enable Remote Access usage.

5.3.8 SVCA

If SVCA is asserted but the associated count is zero, the behaviour is the same as if SVCA was unasserted. Thus, this setting is recommended not to be used.

5.3.9 Digital Export restrictions

Content may be marked as both 'Export beyond Trust' and 'Export Restricted to Controlled CPS'. The behaviour in such a case is exactly the same as if it was only marked 'Export Beyond Trust': Content can be Exported, regardless of any protection or otherwise which may be implemented in the sink device.

NOTE: When Content is marked 'Export beyond Trust', the mapping agreed between a CPCM C&R regime and a Trusted or a Controlled CPS is still to be obeyed.

5.3.10 Uncompleted settings

Content that is marked MLAD, MGAD, VLAD, or VGAD is expected to be movable, copyable or viewable from within the Remote AD once a Remote Access Rule is valid. If no Remote Access Rule is given, content is considered to be MAD or VAD as long as it is Acquired. The content provider is thus advised to do one of the following:

1) Mark Content as MLAD, VLAD, MGAD or VGAD and assert at least one Remote Access Rule.

2) Directly mark the Content as MAD and VAD.

Similarly, if remote_access_date_immediate_flag or remote_access_date_moving_window flag is asserted but no remote_access_date is given, Content is considered to be MAD or VAD from the time of its Acquisition.

If Content is marked VWA but neither a Window-start nor a Window-end is given, then the View-Window restriction is ignored and the behaviour is the same as if VWA was unasserted.

If Content is marked VPA but no View-Period is given, then View-Period restriction is ignored and the behaviour is the same as if VPA was unasserted.

If Content is marked MGAD or VGAD but no geographical restriction is given in CPCM Auxiliary Data, Content is considered to be MLAD or VLAD from the time of its Acquisition. It is actually considered that there was some problem in the USI transmission chain and thus USI are considered to be corrupted.

NOTE: Geographical information may be absent in the CPCM delivery signalling when used but are expected to be present in the CPCM Auxiliary Data after Content Acquisition.

An Acquisition Point receiving content with an uncompleted setting of the USI may directly revert to the corresponding complete setting as described in this clause.

5.4 Unusual settings

5.4.1 Non Viewable

Marking Content as Non Viewable is not contradictory with any setting of Viewing Propagation Information as this only means that Content is still under the control of the source Content Protection System. The source Content Protection System may then set other USI regarding the expected usage once control is transferred to CPCM

5.4.2 Analogue Export restrictions

If 'Export Beyond Trust' is used in conjunction with an analogue Export Restriction, CPCM Instances will enforce this restriction when sending the Content to an analogue output. However, this restriction is unlikely to be enforced once the Content is exported to an untrusted system.

5.4.3 Unusual DNCS settings

The DNCS flag can be asserted in the USI in combination with any other usage restriction. However content providers should be aware that usage rules when used in conjunction with DNCS will be enforced by CPCM Instances while non-CPCM devices may still have access to the Content because it is not scrambled and might not enforce these usage rules.

NOTE: While a CPCM Instance will not deliberately send CPCM Content to a non-CPCM device, there is still the possibility for Content to leave the CPCM protection environment. For example, a non-CPCM application or device may be able to access DNCS-asserted CPCM Content directly from where the Content is stored or from a multicast channel.

5.5 Bit Bucket scenario support

The bit bucket scenario, as described in TR 102 825-11 [i.8], enables a Storage Entity to put AD-bound Content on a non-CPCM storage device such as hard disks or optical media so that it can be retrieved by another CPCM Instance from the same AD.

Scenarios for which bit bucket usage is enabled are those in which the Content licence may be stored under the sole protection of the AD secret. Since other CPCM Instances from the same AD also know the AD secret, they are able to verify and decrypt the Content Licence and can thus enforce it. USI configurations for which the bit bucket scenario is enabled are the following:

- Content is marked CCNA, and
- Content is not marked VPA.

NOTE: The bit bucket scenario is possible for VPA Content once the View Period has started, since the content is remarked as VWA.

Content marked MLocal or VLocal may be stored on a bit bucket, but will not be able to be accessible based upon those permissions and instead will be accessible only upon AD-based permissions. To transfer the Content to another AD, SAC or Device, secret protection is mandatory.

Content marked MLAD or VLAD may also be stored on a bit bucket. However, as long as the proximity restriction has not expired, Content is available only from within the Local AD. Thus, when such Content is accessed from a bit bucket, the sink CPCM Instance verifies first whether the source CPCM Instance (that can be identified using Last_CL_issuer field) is Local using a proximity test. This means that the bit bucket scenario is only possible if the Source CPCM Instance is Local and connected.

The bit bucket scenario is not enabled for VPA Content. However, since Content is re-marked VWA when it is copied or viewed, bit bucket scenario is enabled as long as content is marked CCNA.

Recording CPCM Content on a non-CPCM storage device is always possible. But, in the case where the USI settings do not allow for the bit bucket scenario, CPCM Content may only be retrieved by the Storage Entity that transferred it to the non-CPCM storage. In such cases, the CPCM content licence is actually stored under device secret protection (or under both device secret and AD secret protection,) of the source Storage Entity so that another CPCM Instance may not access to it.

5.6 Corrupted USI

USI can be found either in the Content Licence or, for geographical information, in Auxiliary Data. Thus, corruption of USI may happen only if the full Content Licence or the full CPCM Auxiliary Data are corrupted, i.e. if the signature cannot be verified.

When encountering a corrupted Content Licence or Auxiliary Data, the CPCM Instance should first attempt to obtain a valid replacement Content Licence or Auxiliary Data using the Content Licence Re-Acquisition Protocol described in TS 102 825-4 [i.9]. If this is not available or the replacement Content Licence or Auxiliary Data is also corrupted, the following mitigating actions should be taken.

NOTE 1: There is no way for the CPCM Instance to assess whether the Content Licence Creator field in the Content Licence or the URL field in the Auxiliary Data are corrupted or not. CPCM Instance may attempt to run the Content Licence Re-Acquisition Protocol in the hope that they were not corrupted. The protocol will be successful only in the case they were not corrupted.

NOTE 2: There is no requirement for the Acquisition Point that delivered the content (identified by the CLC) to be able to re-generate corrupted Content Licences.

NOTE 3: There might exist recovery mechanisms outside of CPCM authorised by a CPCM C&R regime.

If the full Content Licence is corrupted, there is no means to decide which part of the USI is corrupted and thus all of them are presumed to be corrupted. Thus, CPCM Instances are supposed to enforce the most restrictive applicable states which are the following in such a context:

- Copy Control Information is 'Copy Never - Zero Retention Asserted'

- Content is not 'Viewable', VWA, VPA and SVCA are not assorted

NOTE 4: The assertion of VWA, VPA and SVCA would not change the behaviour as the Content is neither copyable or movable nor viewable.

- Moving Propagation Information and Viewing Propagation Information are MLAD and VLAD with remote access rule set for moving window with maximal duration as would be set by 'FFFF235959':
  - MLocal and VLocal are not asserted.
  - All exports controls are not asserted and 'image constraint' is asserted.
  - DNCS is not asserted.

Content treated in the above manner is essentially useless and need only be retained in the hope of CL recovery as mentioned above.

If content is marked MGAD or VGAD and associated Content Auxiliary Data is corrupted, Moving and Viewing Propagation Information are considered to be respectively MLAD and VLAD. Other USI are not affected.

6 System

6.1 CPCM Versioning

The CPCM specification is allocated a single version number; 1 for this version of this multi-part deliverable.

If a proposed modification to a given version of the CPCM specification implies some break in interoperability with legacy CPCM implementations, in this case, a new version will be introduced which would describe both the proposed modification and how backward interoperability with legacy CPCM Instances would still be maintained. The new version will be an increment of 1 of the previous. This does not preclude the majority of CPCM structures from remaining the same between versions.

Consequently, CPCM Instances implementing future versions of CPCM specifications are likely to also implement older versions of the specifications in order to maintain interoperability with legacy CPCM implementations. Conversely, this also means that a CPCM Instance implementing this version of the specification may try to communicate and interoperate with CPCM Instances with a higher version number; it is likely that these CPCM Instances will be able to adapt their behaviour and communicate using the lower version protocols.

Creation of a new version of CPCM will also create the need for simultaneously carrying out control structures for different versions of the specification. This is the case for Revocation Lists (see clause 7.8.2.6).

EXAMPLE: A content provider may want to make available its content for versions 1 and 2 of CPCM, though with different permissions as their security level is different. To that end, he can generate one Content Licence with version 1 and one Content Licence with version 2 and attach both versions to the Content. Another possibility is to implement version 2 of the Content Licence as an Extension of version 1 of Content Licence.

NOTE: In the above example, if the Content Provider would rather make available its content with the same permissions for version 1 and version 2 of CPCM, it is sufficient to generate only version 1 of the Content Licence as it will be usable by both versions of the implementations.

If the proposed modification to a given version of the CPCM specification aims at defining new elements rather than modifying existing ones, it is advised to do this by means of a CPCM Extension rather than creating a new version of the specification as this intrinsically guarantees backward compatibility without requiring the simultaneous carriage of different control structures for each version.

6.2 CPCM Data Types

6.2.1 CPCM Identifiers

6.2.1.1 CPCM Instance and CPCM Certificate Identifiers

The role of the CPCM Instance Identifier is to bind the CPCM Instance to the device in which it is implemented. The value is assigned by the hardware or software manufacturer.

CPCM Certificate identifier is assigned by the relevant CPCM C&R regime and is specific to a given CPCM Instance.

Both identifiers are bound together in the CPCM certificate.

6.2.1.2 ADID

ADID is assigned by the CPCM Instance that creates the Authorized Domain. Its first eight bytes are equal to the certificate identifier of the CPCM Instance that created the AD, while the last byte is the index of the AD created by that Instance: the first AD created by that Instance will have index 0, the second index 1 and so on. When 255 is reached, the next index will be zero.

6.2.1.3 Content Licence Identifier

CPCM uses a Content Licence identifier (CLID) rather than a Content Identifier because content is usually managed at a higher-level layer, for example Transport or Storage. The CLID is therefore used to link CPCM information to those upper layers.

CLID is a 128-bit identifier assigned by the Content Licence Creator (see clause 6.5.3).

6.2.2 CPCM Data Structures

6.2.2.1 CPCM date and time

CPCM_date_time conforms to the DVB Service Information specification EN 300 468 [i.10]. The total length of this structure is 40 bits, and it is made up of the Modified Julian Date (MJD) and the Universal Time, Co-ordinated (UTC) field; these are 16 bits and 24 bits respectively.

EXAMPLE: 93/10/13 12:45:00 is coded as "0xC079124500". Algorithms to compute MJD date from actual date and reverse algorithm are described in EN 300 468 [i.10], annex C.

6.2.2.2 CPCM certificate

The CPCM_instance_certificate is the *signed* certificate for any given CPCM instance. This indicates that public signature keys are required to view the information contained. Size of this structure is 2048 bits.

The unsigned certificate is described by the CPCM_instance_certficate_body, for which all the information is open for viewing. Essentially, the types of information held by both structures is identical, the only difference being that one is masked while the other is not. The CPCM_instance_certificate_body is never transmitted but is obtained upon certificate verification.

The CPCM_instance_certificate_body holds all the useful information required for the appropriate handling of a CPCM instance. The total length of the structure is 896 bits. As mentioned above, the CPCM_instance_certificate_body binds the CPCM_instance_id and the CPCM_instance_certificate_id together for a CPCM instance; however as well as these two identifiers, the CPCM_instance_certificate_body also binds together the other characteristics of the CPCM Instance.

In addition to this, there are unallocated bits of the structure which are split as follows:

- 1 bit is reserved and allows byte alignment of the 33rd byte of the certificate. The value of this bit is zero.
- 112 bits for reserved_for_future_use, for use in future versions of the CPCM specification; currently they are set to zero.
- 128 bits for reserved_for_CPCM_extensions, for use by CPCM Extensions and allocated in TS 102 825-14 [i.6]; if unallocated, they are set to zero
- 384 bits for reserved_for_private_extensions, to be allocated by each C&R regime and whose interpretation is governed by the C_and_R_regime_mask field; if unallocated, or if the corresponding CPCM Extension is unsupported, they are set to zero.

6.2.2.3 CPCM certificate chain

The CPCM_instance_certificate_chain is of variable length and may contain up to 15 chained CPCM Instance Certificates (CIC) i.e. up to 15 CPCM_instance_certificate structures. Each certificate contained is the parent certificate of the certificate before; therefore the first CIC is a leaf certificate and the last CIC is the one *signed by the root certificate*. This structure is exchanged during the SAC establishment in order to check the peer certificate.

It contains the following fields:

- certificate_count shows the total number of certificates contained within the chain (4 bits);
- a list of CPCM_instance_certificates containing each signed certificate for the corresponding CPCM instance (2 048 bits).

In addition, 4 bits are reserved for use in future versions of the CPCM specification.

6.2.2.4 CPCM Content Licence

This structure embodies information pertaining to the handling of CPCM Content. It is a variable length structure as it includes two fields which are of variable length i.e. RL_index_list and usage_state_information and two optional fields; auxiliary data digest and AD_secret_signature, whose lengths are also included in the "length" field.

The following method may be followed to parse the Content Licence:

- The first byte is always the CPCM_version. If the corresponding format of the CPCM Licence is unknown (e.g. if the parsing Instance implements only version 1 while CPCM Content licence has an upper version), then the parsing can be stopped at that step.
- The second byte is always set to the length of the Content Licence, excluding the version and length fields (but all other fields are included). If this length does not match the actual Content Licence length, then, the parsing can be stopped. This value is noted as length_CL in the following.
- The third byte is analysed as follows:
  - The most significant bit is content_licence_protection. When it is set to 1, the CL is AD protected. Else it will be protected with the SAC or the device secret. For DNCS content, CL is not protected.
  - The least seven significant bits provide Descrambler_information (see clause 6.2.2.5).
- Bytes 4 to 19 describe the Content Licence Identifier (see clause 6.2.1.3).
- Bytes 20 to 27 describe Content Licence Creator: this value corresponds to the CPCM Instance Identifier of the Acquisition Point that first Acquired the Content Item and is useful when a new Content Licence has to be re-Acquired through that Acquisition Point (see clause 6.5.5).
- Bytes 28 to 35 describe the Last CL Issuer: this value corresponds to the CPCM Instance Identifier of the latest CPCM Instance that modified the CL. If the CL has not been modified since the Acquisition, this value will be set equal to the Content Licence Creator. This field is used when proximity restricted content is recorded onto a bit bucket. When retrieving the Content item from the bit bucket, a proximity check can be performed with the originating CPCM Instance so that proximity restriction is enforced.

- Byte 36 gives the C&R regime mask: this value specifies those C&R regimes which are allowed for that content. In addition, if N is the number of bits set to 1 in this field, then each group of 4 bytes (in bytes 37 to bytes 36+4*N) corresponds to the minimum RL_index of the applicable Revocation List that the parsing Instance needs to possess to enforce the Content Licence. The applicable Revocation List is the one with the same CPCM Version as that of the Content Licence, and which was issued by the C&R regime whose bit is set in the C&R regime mask. Separate RL_index values are given for each bit set in the C&R regime mask, following the order from msb to lsb.

EXAMPLE: If CPCM version is 2 and C&R regime mask is 0x09, then:

  - C&R regimes which were allocated bits 0 and 3, (if bit 0 is the least significant bit,) are authorized for that content.

Byte 37 to 40 is the minimal RL_index of the Revocation List issued by the C&R regime allocated with bit 3 and having 2 as CPCM version. The parsing CPCM Instance needs to have a Revocation List issued by that C&R regime with CPCM version set to 2 and an index not less than the value carried in those bytes.

Byte 41 to 44 is the same minimal index but for the Revocation List issued by the C&R regime which was allocated bit 0.

- Bytes 37+4N to bytes 45+4N give the ADID to which the Content is bound. When set to zero, this means the Content is not bound to any AD.
- Bytes 46+4N to 61+4N give the content descrambling key. If the Content is marked DNCS, value of this field is zero. If the CL is AD protected, this field is encrypted with the AD secret.
- Byte 62+4N is the length of the USI field (noted usi_length in the following).
- Byte 63+4N to byte 62+4N+usi_length give the USI field (see clause 6.2.2.6).
- For the following bytes, the following analysis has to be made:
  - If cl_protection_mode is 0, length_cl may be equal either to 62+4N+usi_length or to 82+4N+usi_length. In the first case, Content Licence parsing is completed and no Auxiliary Data are associated with the Content Licence. In the second case, Auxiliary Data are associated with the CL and the last 20 bytes of the CL correspond to the digest value of the Auxiliary Data.
  - If cl_protection_mode is 1, length_cl may be equal either to 82+4N+usi_length or to 102+4N+usi_length. In the first case, no Auxiliary Data are associated with the Content Licence. In the second case, Auxiliary Data are associated with the CL and bytes 63+4N+usi_length to 82+4N+usi_length of the CL correspond to the digest value of the Auxiliary Data. In both cases, the last 20 bytes of the CL correspond to the signature of the CL using the AD Secret. This signature is computed before the encryption of the descrambling_key field.

6.2.2.5 CPCM Descrambling Information

This is described as a 7-bit field which is applied by the Acquisition Point to communicate which CPCM scrambling parameters have been applied to the CPCM content item. It comprises three 1-bit fields, and a reserved field which is set to zero and is intended for possible use in future versions of the CPCM specification. The most significant bit corresponds to the odd_even_bit_indicator while the last two bits correspond to the scrambling chaining mode and to the MSC mode, respectively.

Some care needs to be taken when populating the odd_even_indicator, particularly when the CPCM scrambling key is being changed by another CPCM Instance before sending it on. The odd_even_indicator has a default setting of "0" indicating the use of an even scrambling key. At the first change of the content key, the new license will toggle the odd_even_indicator bit from its previous setting. In other words, in the case of a default setting of "0", at the first content key change the new content license will have a value of "1" for the odd_even_indicator. Synchronization of these changes with the content scrambling itself is handled by the relevant Adaptation Layer and is described in TS 102 825-9 [i.11].

Dynamic content key changes are described more fully in clause 6.4.4.1.

6.2.2.6 CPCM USI

This field is used to communicate the authorised usage associated with a CPCM content item. It comprises several fields, which together will provide all the copy/move/ view/remote access/export capabilities of the CPCM content item.

This field is variable length. The actual length is given in the first byte of the field. The following 3 bytes are always present and give basic information about authorised content usage. The presence of following bytes is dependent on the actual values of flags in the first bytes. The ordering of these conditional bytes is always the same. Parsing of the USI field is to be made accordingly.

Detailed explanations on the meaning of the different fields can be found in clause 5.

6.2.2.7 CPCM Delivery Signalling

The cpcm_v1_delivery_signalling structure contains CPCM Content Metadata which signals, during acquisition, the Authorised Usage associated with any given CPCM content. The carriage of CPCM delivery signalling is as defined in TS 102 825-9 [i.11].

The identifier contains the same fields as defined in TS 102 825-9 [i.11], and all fields have the same meaning as corresponding CPCM USI field with the following exceptions:

- remote_access_delay_flag (1 bit) indicates that remote access will be permitted after a delay after the Acquisition, as indicated by the remote_access_delay field.
- remote_acess_delay (16 bits) indicates the period of elapsed time since acquisition, after which remote access will be permitted; the time is indicates in increments of 15 minutes.

  EXAMPLE: if the value of that field is 0x0168, content will be accessible remotely 24 hours after its Acquisition.

  NOTE 1: The two above fields are not present in USI as USI are created after Acquisition. In the USI, they respectively correspond to remote_access_moving_window_flag and remote_access_date.

- remote_access_date_flag (1 bit) indicates that remote access will be permitted after the date which is indicated by the remote_access_date field.

  NOTE 2: This bit corresponds to the bit remote_access_date_immediate_flag in the USI.

- remote_access_date (40 bits) indicates the date after which remote access will be permitted; this structure follows the structure of CPCM_date_time i.e. it is made of Modified Julian date (16 bits) and Universal Time, Co-ordinated (24 bits).

  NOTE 3: If the start date is undefined, then all bits are set to 1.

How these fields are mapped to USI is described in clause 10.3.

6.2.2.8 CPCM Status information

The CPCM_status_information structure is used to communicate the status of a particular CPCM Instance, as a result of a request from another CPCM Instance. It contains:

a) static information which is duplicated from the CPCM Instance Certificate; and

b) the ADID. ADID value is set to zero if the CPCM Instance is not member of an AD.

It is used in the CPCM Instance Status Enquiry protocol and in the CPCM Discovery protocol.

6.2.2.9 CPCM handling operation

CPCM Content handling is performed only by the CPCM Instance inside the device. Content Handling within a CPCM instance is defined by the use of one or more of the five CPCM content handling operations referred to collectively as APECS, Acquisition, Processing, Export, Consumption, Storage, that are necessary to fulfil a device's intended functionality.

The CPCM_content_handling_operation structure is used to signal the application of one or more of the APECS operations. It is represented by a one byte operation, with 3 reserved bits, and five operational bits. The semantics for each bit assignment are as follows:

- Acquisition: bit set to 0 if no Acquisition operation is indicated and set to 1 when an Acquisition operation is indicated.
- Processing: bit set to 0 if no Processing operation is indicated and set to 1 when a Processing operation is indicated.
- Export: bit set to 0 if no Export operation is indicated and set to 1 when an Export operation is indicated.
- Consumption: bit set to 0 if no Consumption operation is indicated and set to 1 when a Consumption operation is indicated.
- Storage: bit set to 0 if no Storage operation is indicated and set to 1 when a Storage operation is indicated.

This field is used for instance when Content Licence is transmitted to indicate what the Content item is Requested for or what is the authorized operations on the Content as authorizations may vary regarding the operation that is to be performed on the content.

EXAMPLE 1: If the CPCM Instance is a Storage Entity and a Consumption Point, its CPCM Content handling Operation will be 0x03.

EXAMPLE 2: If content has a limited number of permitted simultaneous views, it is necessary for the Acquisition Point delivering the Content Licence to know the purpose of the request. For instance, if the Content is transmitted to a CPCM Instance that is a Storage Entity and a Consumption Point, the Acquisition Point needs to know whether the content is requested for Consumption or for Storage as this will affect its count. It also needs to know the final destination of the Content when the request comes from a Processing Entity, as the content will be further transmitted after its has been processed.

6.2.2.10 CPCM Revocation List

This structure carries a list of CPCM Instances and ADs that have been revoked by one C&R regime for a given CPCM version.

The structure embodies the following fields:

- CPCM Version.
- RL_index, which is an index incremented by 1 at each new RL release. It is used for an Instance to check whether a Revocation List has been updated. It is also used in CPCM Content Licences to specify the minimal RL_index a CPCM Instance should know to access the Content. RL_index incrementing is unique to a given combination of a given CPCM version and a given C&R regime mask. In the case where there are several C&R regimes for the same CPCM version, RL_index will be incremented independently for each C&R regime. A C&R regime will re-start with an RL_index of 0 when issuing the first Revocation List relative to a new CPCM version.
- C&R regime mask which specifies by which C&R regime the Revocation List was issued. Exactly one bit is set in this mask in a Revocation List.
- Certificate generation index: This field gives the minimal generation index of the certificate in order not to be viewed as revoked.
- Certification generation index last change: This field provides the list of RL_index corresponding to each change of the certificate generation index in the Revocation List. For certificates that are revoked because of the certificate generation index mechanism, it is possible to determine what the revocation index was when it was revoked and consequently to determine whether it may access the content or not as described in clause 6.4.7.
- The count of revoked certificates.
- The count of revoked ADs.

- The corresponding list of revoked certificates.
- The corresponding list of revoked ADs.

The structure has a signature that can be verified using the relevant C&R regime revocation public key.

6.2.2.11 CPCM Revocation List Locator

CPCM enables the discovery of available revocation lists. However, actual transmission of the lists is done by the HN layer. This field enables the HN layer to locate where the Revocation List can be found.

The first byte of the field corresponds to the length of the field, first byte excepted. The interpretation of the following byte is dependent upon the HN layer and defined in TS 102 825-9 [i.11].

6.2.2.12 CPCM Auxiliary Data structures

6.2.2.12.1 CPCM Auxiliary Data

The CPCM_auxiliary_data structure is used to carry CPCM Auxiliary Data. The CPCM Auxiliary Data structure is securely bound to the Content License but is transported independently. CPCM Auxiliary Data integrity is guaranteed through the Content Licence to which it is bound: The Content Licence is itself signed and carries the digest value of the Auxiliary Data. Any modification of the CPCM Auxiliary Data results in a modification of the Content Licence body and thus a new signature has to be re-computed for the Content Licence. This can only be performed by a CPCM Instance.

Independent transport of CPCM Auxiliary Data allows for independent verification of the Content Licence, which is more efficient. Furthermore, data comprised in the CPCM Auxiliary Data is used less frequently than data carried in the Content Licence. Verifying the integrity of Auxiliary Data is only mandatory when values carried therein are used. This improves implementation efficiency.

Once CPCM Auxiliary Data integrity has been checked, implementations may retain this information to avoid performing the verification again. In this case, retained data is secured against unauthorized modification and is strictly associated with the verified Content Licence.

CPCM Auxiliary Data includes a header which comprises two elements; the content_licence_identifier (CLID) which identifies the CPCM Content Licence with which the CPCM Auxiliary Data is associated; and CPCM_auxiliary_data_element_counter which identifies the total number of CPCM auxiliary data elements.

Additionally, it includes a body embedding the actual CPCM auxiliary data elements. Each CPCM auxiliary data element is composed of the following items:

- CPCM_auxiliary_data_element_identifier is the identifier of the element carried.
- CPCM_auxilary_data_element_length which indicates the length in bytes of the element, but only for variable-length elements. Note that this field is omitted for fixed-length elements. Element_length is mandatory for variable length elements and for any new element which may be defined in a new version of the CPCM specification.
- CPCM_auxiliary_data_element, which defines a CPCM Auxiliary Data element.

If no auxiliary data element needs to be included in the CPCM Auxiliary Data, the whole structure can be omitted. The corresponding digest in the Content Licence is in this case also omitted.

This structure may also be used by CPCM or private extensions to include the information they require.

6.2.2.12.2 CPCM geographic area format

The CPCM geographic area format structure carries the geographic area to which MGAD or VGAD restriction applies. It carries a list of geographic locations (see next clause), and comprises two elements: location_count which is the number of CPCM geographic locations in the list, and CPCM_geographic_location_information which specifies the location information.

6.2.2.12.3 CPCM geographic location information

This field has two usages: It can be used within the CPCM geographic area structure. It is then embedded in CPCM Auxiliary data and indicates the geographic location for which the Content is authorized. The second usage is in the geographic information protocol to transmit the current geographical information.

The CPCM geographic location information comprises three data elements:

- geo_location_format which specifies which kind of location information is given (see below);
- geo_location_element_length, indicating the length in bytes of the geo_location element for variable-length elements (e.g. URL based location information).This field is omitted for fixed-length elements. Element length is mandatory for variable length elements and for any new element that may be defined in a new version of the present document; and
- thirdly, geo_location which gives the actual geographic location.

Four kinds of geo_location are possible; firstly a 2-byte alphanumerical country code, which specifies a country corresponding either to the current geographical location or to an authorized location when used in the CPCM geographic area structure, secondly country code (excluded) which is a 2-byte alphanumerical country code which is used in the geographic area structure to exclude remote access to the CPCM content from within the indicated territory or territories. It is not used when transmitting the current location, thirdly 2D absolute position which **is** a coding of an absolute geographical position in terms of latitude and longitude. It is only used when indicating the current location and is not used to signal any Authorized Usage for CPCM Content, and finally URL pointing to location which is a URL that points to a web page from which the CPCM_geographic_area structure can be read. The URL can be used to indicate the current geographical location or it can be used within the Auxiliary Data.

6.2.2.12.4 CPCM geographic location formats list

The CPCM_geographic_location_formats_list structure carries a list of geographic location format identifiers. It comprises two data elements which are; format_code_count containing the number of CPCM geographic format codes contained in the list, and CPCM_geographic_location_format_code.

This field is used in the CPCM geographic information protocol to specify which format codes are actually known to a given CPCM Instance.

6.2.2.12.5 Other CPS Export data

The other_CPS_export_data structure carries data that is specific to another CPS to which CPCM Content can be Exported. CPCM Instances that do not implement the indicated other_CPS will ignore the associated private data. It comprises three data elements: CP_system_id, which identifies the other CPS to which the private data applies, other_CPS_export_data _length which defines the length of the other_CPS_export_data, and char which is an other_CPS export data byte.

EXAMPLE: This element may be used to carry the relevant USI mapping to be applied in the case where the content is to be Exported to the CPS defined by the CP_system_id.

6.2.2.12.6 CPCM rights issuer URL

The CPCM_rights_issuer_URL structure carries the URL (Uniform Resource Locator) where additional Authorized Usage rights for the associated Content Item can be obtained. It comprises two data elements which are; URL_length giving the length of the URL expressed as the number of bytes contained in the URL, and char which is the UTF-8 coded character of the URL.

It may be used in super-distribution scenarios to locate where rights may be purchased for the content or to locate an Authorized Authenticated Agent which can allocate additional rights for the content.

6.2.2.12.7 CLC private data

The CLC_private_data structure carries data that is specific to a proprietary system, for example a Conditional Access system from which the CPCM Content Item was Acquired. CPCM Instances that do not implement the indicated CA system will ignore the associated private data. It comprises three data elements which are; CA_system_id which identifies the CA system to which the private data applies; CLC_private_data _length which defines in bytes the length of the CLC_private_data, as the number of bytes, and char which is the CLC private data byte.

This field may be used by the CA or the DRM that delivered the content to record information about the content that may be later used if, for instance, the user requests additional rights for the content. Another possible usage is for the CAS/DRM to store proprietary information when the content has been Acquired but not yet purchased, i.e. is marked as 'Not Viewable'.

6.2.2.12.8 Key Recovery Information

This field carries proprietary information allowing the relevant Authorized Authenticated Agent to recover the Content Scrambling Key. This field has two elements, which are:

- key_recovery_information_length which carries the byte length of the key recovery information;
- key_recovery_information which carries the actual recovery information.

The encoding and the interpretation of this information are proprietary and hence not defined in this multi-part deliverable.

This field can be used in the super-distribution scenario as described in TR 102 825-11 [i.8].

The following gives a non-exhaustive list of examples as to how this field could be used:

- This field could carry the content scrambling key encrypted with a public key of the Authorized Authenticated Agent.
- This field could carry a seed allowing the Authorized Authenticated Agent to re-generate the content scrambling key.
- This field could carry the identifier of the AP that generated this data together with the content scrambling key encrypted using a key known the AP. The identifier will be used by the Authorized Authenticated Agent to regenerate the key known to the AP and then to recover the content scrambling key.

6.2.2.12.9 External Scrambling Information

This field may be used in the case of a smart card based Acquisition Point wherein the smart card embeds a CPCM Instance, as explained in TS 102 825-2 [i.4]. The usage of this field allows the smart card to transmit to the CPCM Instance within the host such information as may be necessary to descramble the content and possibly to instruct which CPCM LSA mode and which CPCM scrambling key to be used to CPCM scramble the content.

When used, this field will be incorporated in the Auxiliary Data and the associated Content Licence will then require SAC protection. Key fields of this structure will be also encrypted using the SAC Secret. This field may only be interpreted by an Acquisition Point. Hence, if there is no requirement to re-issue a new Content Licence and Auxiliary Data with this field removed, it is still recommended to do this in order to delete information relating to the external scrambling.

This field includes the following elements:

- Total length of the field.
- Whether the CPCM scrambling key is included, and if so, which LSA mode needs to be used.
- Whether one or two external scrambling keys are included. The inclusion of a second scrambling key allows smooth transition between crypto-periods.
- Algorithm that needs to be applied to descramble the content, if known to the smart card. If not known, 'unspecified' is to be used and the host application will have to obtain this information in a different way.

- If the algorithm is AES based, the specific AES mode that needs to be used to descramble the content. This information is in the external_scrambling_algorithm_option field and is ignored for non AES ciphers.
- The length of the external scrambling key field. This information is necessary when the algorithm is unspecified. In all other cases, it may be used to check that the transmitted information is consistent.
- The external scrambling key field that embeds one or two (if the additional_external_scrambling_key flag is asserted) scrambling keys needed to descramble the content.
- If applicable, the CPCM scrambling key that is to be applied to CPCM scramble the content.
- Optionally, a variable-length proprietary field whose interpretation depends on the external system. The presence and length of such a field are deduced from the total length of the external scrambling information element.

6.2.2.13 Authorised Domain structures

6.2.2.13.1 AD name

AD name is a string of up to 31 UTF-8 characters, aiming at giving a human-readable name to the domain, contrarily to the ADID which is chosen by the CPCM system regarding specific CPCM parameters. It is configurable by the user.

The first byte of the field is the number of bytes (always less than 32) of the rest of the field.

6.2.2.13.2 AD capability

This 1-byte field is used to characterize a CPCM Instance capability with regards to AD management.

The different capabilities are:

- Nothing: all the bits are zero. In this case, the CPCM Instance is not able to do any operation linked to AD Management. This includes also enforcement of AD bound content. Such a CPCM Instance will not be able to transmit CPCM content, except when the content is marked MCPCM or VCPCM or when the content is marked MLocal or VLocal.
- AD aware: In this case, value of the field is 0x10 or 0x11 if the Instance is also ADSE countable (see below). Such an Instance is not able to perform any AD Protocol, as described in TS 102 825-7 [i.12] but can be an AD member using other means, if allowed by a CPCM C&R regime, and can thus enforce Movement and Viewing of CPCM AD-bound content.
- ADM capable: In this case, value of the field is 0x18 or 0x19 if the Instance is also ADSE countable. Such an Instance is able to run all AD protocols supported by the applicable C&R regime as an AD member. It is not capable of becoming a Local Master or a Domain Controller.
- LM capable: In this case, value of the field is 0x1C or 0x1D if the Instance is also ADSE countable. Such an Instance is able to run all AD protocols supported by the applicable C&R regime as an AD member and as a Local Master. It cannot become a Domain Controller.
- DC capable: In this case, value of the field is 0x1F as the CPCM Instance is necessarily ADSE countable, if the C&R regime implements the ADSE method proposed in TS 102 825-7 [i.12]. Such an Instance is able to run all AD protocols supported by the applicable C&R regime in any role.
- ADSE countable: In this case, value of the field is 0x11, 0x19, 0x1D or 0x1F, depending on its other abilities. Such an Instance will have to join and leave an AD with a Domain Controller and will be counted in the different AD counts.

6.2.2.13.3 ADSE values and DC ADSE values

ADSE_values is a 20-byte structure that carries the different counts relative to a Domain Controller in an AD. Different ADSE_values corresponding to the different Domain Controllers of the AD are listed in the structure named DC_ADSE_values, which consists of a first byte carrying the number of Domain Controllers in the list followed by a list of the CIC identifiers of each DC followed by its current ADSE values.

The ADSE_values structure is maintained by each Domain Controller. Implementations have to protect its integrity and prevent the replaying of old values. ADSE_values are also stored in the AD_internal_record, which is maintained by each Domain Controller. It is also used in DC Transfer, Split, Merge and Rebalance protocols to carry the ADSE values which are related to a single DC.

The DC_ADSE_values structure is used in various discovery and AD management protocols to provide information regarding the remaining counts of each Domain Controller. It is also used in the Local Master Election process to determine which Domain Controller is preferred to become Local Master.

6.2.2.13.4 DC Local identifiers

This structure carries a list of CPCM Instances Certificate identifiers that are or have been Domain Controllers. The first byte of the structure carries the number of identifiers in the structure and is followed by a corresponding number of CIC identifiers.

This structure is uniquely used in the AD internal record to carry the identifiers of the current DC(s) in a single AD or the identifiers of those DCs which have merged.

6.2.2.13.5 AD internal record and AD internal record list

The AD internal record carries AD information relative to one DC. It carries:

- the ADID,
- the AD name,
- the DC CIC identifier,
- the DC local identifier, which is a short identifier allowing the identification of each DC within the AD,
- the current ADSE values,
- the list of current Domain Controllers,
- the list of DCs that have been merged, and
- optionally, the last change date of the AD.

The AD internal record list is a structure carrying a set of one or more AD internal records associated with a single AD.

Usage of these structures is given in clause 8.3.4.

6.2.2.14 CPCM Extension and private elements structures

CPCM is a flexible system that can be extended both in a standard way and in a proprietary way. When it is necessary for a CPCM or a private extension to have additional data, two possible structures are available. If the extension forms part of the CPCM specification, such as Playcount (TS 102 825-14 [i.6]), then the CPCM extension data element structure is used. If the extension is private, i.e. does not form part of the CPCM specification, then the Private element structure is used.

The CPCM extension data element structure allows data to be used by a CPCM Extension.

The structure comprises a 16 bit CPCM_extension_identifier, which identifies the CPCM Extension. Extensions are specified within TS 102 825-14 [i.6].

EXAMPLE Play Count is represented as 0x0001, followed by a 16 bit CPCM_extension_data_length which gives the length of the CPCM Extension data, and finally the CPCM_extension_data which is the CPCM Extension data itself.

In a similar manner, the Private element structure allows data to be used by a non-CPCM Extension.

This structure, if present, allows specific implementations to add message elements or CPCM Auxiliary Data elements that are not part of the CPCM specification.

The structure comprises a 32 bit private_element_identifier as specified in EN 300 468 [i.10]. Allocations of this field are defined in TS 101 162 [i.13]. This is followed by a 16 bit private_element_data_length specifying the length of the private data, and finally private_element_data, which is the private data itself.

NOTE: Allocation of private_element_data_identifier is made on a per entity basis. If an extension needs to define different elements to be inserted in the Auxiliary Data or in CPCM messages or if an entity defines several private extensions using different private elements, it will need to define its own private element syntax allowing implementations to define rapidly which elements are relevant to them, as all these elements will have the same identifier.

6.3 CPCM Protocols

Protocols in CPCM define the language and structure of the "conversation" taking place between two CPCM devices.

The protocols are describes in terms of the messages, their contents, and the logical state machines that govern the flow of the conversation. Implementations are not constrained in how to write software for these protocols, provided that the behaviour of signals "on the wire" between devices conforms to the CPCM System Specification.

6.3.1 CPCM Message Framework

CPCM messages are defined in a way that is independent of the underlying network. This means that implementations are able to using existing messaging infrastructures where available, such as UPnP, or otherwise send the CPCM-defined messages directly using whatever method is available. This may be as complex as a structured message routing system, or as simple as bytes on a serial connection (e.g. between a smart card and its host). The CPCM specification describes the appropriate mapping of CPCM messages onto a number of different network types.

Whatever technology is employed, it is important that the messaging infrastructure is to some degree multi-threaded, such that the execution of CPCM software is never "blocked" while awaiting the arrival of a CPCM message. Otherwise there is a risk that a delayed or lost response may "freeze" the user experience.

Some CPCM messages are "broadcast". This means that they are delivered almost simultaneously to multiple devices on the reachable network. This may in fact use a broadcast mechanism in the underlying message transport such as Ethernet broadcast, or it may be implemented using multiple individual message deliveries such as with UPnP.

6.3.1.1 CPCM Protocol Message

CPCM messages follow a common conventional structure, designed both for flexibility and for efficient processing in devices with limited memory and processing power.

While the specification provides documentation of the messages in both "API function call" form and a description of the binary structure, it is not a requirement to implement the function calls in software. It is only necessary that the format of the message "on the wire" is correct.

CPCM Messages can be encrypted, signed or both. In this version of the multi-part deliverable, message encryption is only done using the SAC session key. Encryption is applied to the message payload only. It is neither applied to the message header nor to the message signatures. Signatures are always computed over the clear message, including the message header. Messages can be signed using the SAC session key, the AD Secret, or both. In the latter case, The SAC session key signature is first computed and then appended to the message. Then, the AD Secret signature is computed over the full message, including the SAC signature, and similarly appended. Computing the AD Secret signature over the SAC signature allows for ensuring that the signing Instance actually knows the AD Secret, as this signature cannot be replayed. This would otherwise not be the case, as the AD Secret signature would be computed over a constant field.

Auxiliary Data are never part of this signature or encryption process and are always appended after the last signature field. Some fields may be separately encrypted, if they require it. Authentication of the Auxiliary Data is achieved thanks to the presence of the Digest in the Content Licence.

NOTE: It is possible to include the Auxiliary Data in a message without including the Content Licence. In this case, they will not be authenticated and thus may only be used for information and not for enforcement purposes.

6.3.1.2 CPCM Protocol Timeouts

Communication protocols generally employ timeouts has a means to determine that a message has been lost, or that the other end of the communication failed to respond.

TS 102 825-4 [i.34] does not define specific values of any timeouts. These timeouts may possibly be defined by a CPCM C&R regime or by the underlying network technology.

In the absence of any definition, the implementer may choose its own timeout parameters, with the following guidelines:

- CPCM is a network agnostic technology and works with heterogeneous network topology. Thus, even if the CPCM Instance uses a performing network technology, the path to the destination CPCM Instance may go through slower network links.
- CPCM works also over Wide Area Networks. This could be done e.g. through a VPN connection which could cause extra delays.
- A too short timeout may result in re-sending the message before the reply was received and cause troubleshooting. A possible way to overcome this is for the destination CPCM Instance to ignore any duplicate CPCM message before the timeout of its reply has itself expired.

That said timeout values have a direct impact on UI responsiveness, so the following points should be born in mind:

- When a CPCM instance is waiting for a response, some indication should be made to the user that a process is underway. This could be an hourglass icon, a flashing light, or a progress bar or some similar indications. This reduces the chance that the user interrupts a valid process too early.
- On an ordinary Ethernet or Wi-Fi based home network, a timeout of between 5 and 10 seconds might be appropriate.
- Devices connected over slow serial links, such a Smart Cards, will probably need to be more delay tolerant.

6.3.1.3 CPCM Protocol Bridging

In some cases there is a need for CPCM messages to be forwarded between two interfaces. This could be where a point-to-point connection, such as a serial link, is used between two devices, and a device beyond that connection needs to communicate with a device other than the one it is directly connected with. Another case could be where the first device is not aware of the correct network address for the device with which it needs to communicate.

In general, where a CPCM implementation receives a CPCM protocol message that should go to another known device that may not be receiving it directly, the implementation should relay the message over the appropriate link to the unreached instance.

When a CPCM Instance is providing protocol bridging, it is not necessary that it is able to interpret the messages being relayed. Provided that the bridging instance is able to determine the destination of the message, it should be relayed irrespective of whether the message can be parsed or not.

EXAMPLE: A CPCM device is connected to a home network, and also includes a socket for a CPCM-enabled smart card. A broadcast CPCM Discovery message is received from another device on the home network. The host receives the message, and also forwards a copy to the instance on the smart card via the smart card serial link.

6.3.2 CPCM Protocol Errors

The generic error reporting mechanism is important for aiding both developers of new CPCM software, and for giving meaningful feedback to users and technical support staff during abnormal operation.

6.3.2.1 General Points

In many cases, the protocol error reporting is for the benefit of another device than the one which identifies the error condition. It is therefore important that the error reporting is used correctly and efficiently to provide maximum information about the circumstances.

There are a large number of possible error states that can be reported using this mechanism, and it may not always be obvious which error code to use. Some general guidance is included below.

6.3.2.2 Unspecified error conditions

It is important to avoid the temptation to use this error code too often. It is very unhelpful for developers, consumers and support centre staff to see "Unknown Error" messages. Please confine the use of this code to situations where there really is no better code to use.

6.3.2.3 Message Syntax error

This code should be used only when the structure of the message is incorrect, for instance if the length values do not add up or the format is clearly wrong. It should not be used for invalid field values, if the structure of the message seems otherwise correct.

6.3.2.4 Field value out of range

Use this error code when a message field value is clearly incorrect. This may be because the value is outside the range permitted in the specification, or because it is out of range in the current operational context.

6.3.2.5 Protocol context error

Use this error code when a message is received that is recognised, but not expected in the current state for this transaction. The generation or receipt of this message should normally be treated as terminating the affected transaction unsuccessfully on both ends.

6.3.2.6 Function not implemented

Use this error code for situations where the implementer has chosen not to implement an optional CPCM feature. It may also be used when an unrecognised protocol message type is received, as this situation may indicate an incorrect CPCM version being signalled.

6.3.2.7 CPCM System version error

Use this error code when an instance receives a CPCM message marked as being of a later version than it supports. A message from an earlier version should never be rejected, as updated instances should always recognise the "legacy" protocols for reasons of interoperability.

6.3.2.8 Protocol message SAC signature error

Use this error code when the SAC signature does not match the message content. In some few cases it may still be possible to interpret the message usefully, for instance if the corrupt message is itself an error message, however in the vast majority of cases this message should terminate the transaction unsuccessfully on both ends. In any case, the contents of the message should be treated as untrustworthy.

6.3.2.9 Protocol message AD signature error

Use this message when the AD signature does not match the message content. Sending or receipt of this error code should terminate the transaction unsuccessfully on both ends; however it does not invalidate the ADID.

NOTE: If the message is both signed with the SAC secret and the AD secret, 'Protocol message SAC signature error' should be preferably generated.

6.3.2.10 Protocol timeout

Generate a message with this error code when a state machine timeout is exceeded. Unlike most error codes, this one can arise when there is no received message. In such a case, the CPCM_protocol_message_type_request should instead be set to the message type of the message that was sent and which usually started the timeout which has now expired.

NOTE: In case of timeout, it is advised to try re-sending the previous message once or twice before aborting the protocol. This mitigates occasional loss of messages that could be caused by bad network conditions such as Wi-Fi connection loss.

6.3.2.11 SAC expired

Use this error code when a message is received with SAC encryption or signature, for which the associated SAC is no longer valid. This may not be possible if the implementation is unable to remember expired SAC sessions due to memory restrictions or other concerns. In such circumstances, it is acceptable to use the "SAC not established" error code instead.

6.3.2.12 SAC not established

Use this error code when a message needs to be sent that requires carriage using SAC signature or encryption, but a SAC has not yet been established. This error code should also be used when an incoming message is signed or encrypted using a SAC which does not seem to exist. This may be because the SAC fields are corrupt, or because a previous SAC has expired and been forgotten by the receiving instance. When a message is received that is signed or encrypted using a known but expired SAC, the "SAC expired" error code should be used instead.

The error code may also be used when SAC needs to be established to run a protocol completely, even when the first message of the protocol does not actually require SAC protection.

EXAMPLE: SRTT proximity test only protects the last protocol message with the SAC secret but the tested CPCM Instance may generate this error when receiving message SRTT_request, anticipating that the protocol will not be able to complete the test properly.

6.3.2.13 Destination CPCM Instance not present

Use this error code when a message is directed to a CPCM Instance that does not currently exist in the receiving device. This may arise if, for example, a CPCM instance previously existed in an application which has since been shut down.

6.3.2.14 Message protection error

Use this error when the message protection, such as encryption, SAC signature and or AD signature, does not conform to the CPCM specification.

6.3.2.15 Unavailable CPCM Instance

Use this error when the CPCM Instance is busy and cannot reply to the request immediately.

EXAMPLE: The Sink CPCM Instance is a Domain Controller engaged in a DC Transfer transaction. It might decide not to reply to DC-related requests as it may shortly no longer be a DC.

6.3.2.16 Certificate is revoked

Use this error when a SAC has been established while at least one certificate was revoked and the request requires using the SAC Secret for an action that is not permitted.

EXAMPLE: An ADSE countable CPCM Instance is requested to perform a Quorum Test by a revoked CPCM Instance and a SAC has already been established between the two Instances. As this operation is not content management related, it generates this error in reply to the Quorum Test. The same CPCM Instance as above requests a Content Licence that requires SAC protection and for which the requesting CPCM Instance is revoked. The error is also generated. Finally, if the CPCM Instance is requested a Content Licence for which the requesting Instance is not revoked no error is generated and the Content Licence can be sent under the SAC protection.

6.3.3 Transaction Protocols

The use of the Generic Atomic Transaction approach is necessary to ensure that the system can always be returned to the original state in the event that communications break down or there is an error that prevents completion of the operation. Thus, the transaction is atomic into that it is all or nothing. Such a transaction should always be fully completed or otherwise fully abandoned, with no loss of security or resources (rolled back).

Implementers should consider developing a single reusable state machine for handling the various CPCM transactions based on the generic atomic transaction. However this approach is not essential.

Transaction protocols are used for the Content Licence Move Protocol and for six ADM protocols; AD Joining, AD Leave, Domain Controller Transfer, Split, Merge and Rebalance.

Transaction Protocols consist of five messages exchanged between the Content Handling (for the CL Move protocol) or the ADM (for the six ADM protocols) parts of the CPCM Instances and two messages exchanged between the security controls. Messages Begin, Commit and Finish are sent by the client CPCM Instance while messages Ready and Confirm are sent by the server CPCM Instance. Messages between the security controls are exchanged between the sending of messages Commit and Confirm.

6.3.3.1 (Secured) Transaction rollback

This message is used when one party in the transaction protocol stops the protocol following an action from the user or a permission denial from the Security Control, or another error. It informs the other party that the transaction is abandoned and thus reserved resources may be freed, and stored data may be erased. These messages may be used up to a given point in the protocol. After that point, resources have already been committed and the protocol has to be completed. Another protocol will then have to be run to undo the changes.

A transaction rollback message is used if a SAC has not been put in place; else a secured transaction rollback message is used. Securing the message in this way prevents any other entity from cancelling the protocol, and thus mitigates denial of service attacks. Transaction rollback messages coming from an Instance with which no transaction protocol is run, or with which a SAC is established, are therefore ignored.

6.3.3.2 CL and ADSE counts management

Transaction protocols involve the transfer of a right or a change of an ADSE count. These changes are managed so that it is not possible to gain any new rights by interrupting the protocol in its course, in that the right is first removed or the ADSE count is first decremented before the corresponding gain is actually credited.

Transaction protocols can be split in the two following main categories:

Protocols for which the client Instance has gained capabilities or rights after the execution protocol (and hence the server Instance has lost some). This is the case for the AD Join, the DC Transfer, the DC Split and the DC Rebalance protocols. For all these protocols, the server decrements its capabilities and changes its AD status upon receiving the message Commit from the client while the client gains its capabilities and changes its AD status upon receiving the message Confirm from the server.

Protocols for which the server Instance has gained capabilities or rights after the execution protocol (and hence the client Instance has lost some). This is the case for the AD Leave, the DC Merge and the CL Move protocol. For all these protocols, the client decrements its capabilities upon sending the message Commit from the client while the server gains its capabilities upon receiving the message Finish from the server. In addition, if the AD status of the Instance changes, this is only done upon receiving message Confirm. This enables restarting the protocol while the server Instance does not detect any status change and thus does not erase any CIC identifier (see clause 6.3.3.4.1).

6.3.3.3 Abnormal Behaviour

The behaviour described in TS 102 825-4 [i.9] corresponds to a normal running of the protocol. However, many problems or interruptions may occur:

- Network problems or Attacks causing the loss or corruption of some of the exchanged messages, or the addition of new messages. The following describes some of the possible behaviour that may be adopted in such cases:
  - If no response message is received, the CPCM Instance may try to re-send the message several times.
  - If a faulty message (i.e. badly formed or protected) is received, an error is generated. It is recommended that the CPCM Instance accepts messages that may be resent in response to the error message or any other message sent by the legitimate CPCM Instance (in the event of the first received message being erroneous).

EXAMPLE: A faulty message Ready may be sent by an attacker and received by the Source Instance before the genuine message Ready. In this case, the Source Instance generates an error in response to the first message but deals with the second as if the first message was never received.

  - If an error message is received, and if this error message is not related to a refusal of a user or to an impossibility to proceed (e.g. because of security issues), the CPCM Instance may try to re-send the original message.
  - Messages received out of the context of the protocol (e.g. if message Finish is received before message Confirm was sent or if message Confirm is received before the any message has been exchanged between the security controls) are ignored, after a generic error message has been issued.
  - When the same message is received several times consecutively, only the first one is taken into account.
  - Rollback messages are ignored if a SAC exists with the sending CPCM Instance. When the SAC does not exist, and both the expected message and a Rollback message are received, only the expected message is processed; this avoids easy denial of service attacks.
  - When a secured Rollback and an expected protocol message are received, the Instance deals with them in the order of reception. This situation occurs for instance if the user cancels the protocol on a device that has just sent a protocol message. It may also occur if some messages are maliciously replayed from a previous session while the SAC has not been renewed. It is therefore recommended to renew the SAC each time a new protocol is started.
- The user may deliberately decide to stop the protocol while it is running. As this protocol is fully reversible, it is advised to:
  - Accept any cancellation on the client device as long as the client security control has not replied to the request of the server security control. If this cancellation is received, the source Instance immediately sends a secured Rollback message.
  - Accept any cancellation on the server device before sending the security control message. If this cancellation is received, the sink Instance immediately sends a secured Rollback message.
  - Accept on the client device a Secured Rollback message until a request message from the security control has been received.
  - Accept on the server device a Secured Rollback message until a response from the security control has been received.

When a cancellation is accepted, CPCM Instances erase all recorded information and revert to their original state.

Cancellation requests on the device or secured rollback messages received outside of the above windows are ignored.

6.3.3.4 Failure Recovery

6.3.3.4.1 Recovery cases

Transaction protocols have been designed to be able to restart when interrupted. The restarting mechanism is based on the recording by the source and sink security controls of the CPCM Instances of the CPCM CIC identifier of the peer CPCM Instance. Thus, protocol interruption leads to one of four different cases:

- Both source and sink CPCM Instances have no recorded CIC identifiers. In this case, this means that the source CPCM Instance has not yet sent the Commit message and, obviously, the sink CPCM Instance did not receive it. Resources have not yet been reserved. Thus, all the reserved resources will be freed when the CPCM Instance acknowledges its peer will not answer, and protocols will be able to restart without any resources loss.
- The source has recorded its peer CIC identifier but the sink has not. In this case, the message Commit was sent but was not received.
- The source and the sink have both recorded their peer CIC identifiers. This may happen between the sink Instance receiving message Commit and the source instance sending message Finish.
- Only the sink instance has recorded its peer CIC identifier. This may happen if the message Finish was sent but was not received.

Actions to restart the protocol are only needed in the last three cases.

There are no means for a CPCM Instance to know the state of its peer. Thus, from the source CPCM Instance perspective, the second and third cases are similar. So, whenever it has a recorded CPCM instance identifier, upon reconnecting to a network, it runs the discovery protocol and tries to see whether its peer Instance is connected or not. If it was running a DC Transfer, Split, Merge or Rebalance protocol, it is DC capable and runs the Discovery protocol so that it also discovers remote Instances. If it discovers the CPCM Instance whose identifier was recorded, it sends the relevant message Begin in order to restart the protocol from the beginning. If the protocol was a CL Move, an AD Join or an AD Leave, it also restarts the protocol with the Local Master, inserting in the delegation field of the message the recorded CPCM identifier to indicate to the Local Master that the protocol is to be run with this CPCM Instance. Upon restarting the protocol, implementations are advised to inform the user, and to propose protocol cancellation, if it is still possible (see clause 6.3.3.2).

If it does not discover the relevant CPCM Instance, it may decide to run the protocol with another CPCM Instance, in which case some credentials may be lost. Implementations are thus recommended not to run any new protocol unless having sufficiently informed the user of the probable consequences.

This covers the two cases where the source CPCM Instance has a recorded CPCM identifier. The remaining case is when only the Sink has a recorded CIC Identifier. This happens only if the message Finish has been issued but not received.

For an ADM protocol, Sink CPCM Instances with a recorded identifier can send an ADM Invite message with protocol status set to 1 when discovering the Source Instance. If the Source Instance replies with error 'Protocol cannot be resumed', this means that it has erased the recorded identifier and thus that the protocol was completed. The Sink Instance then behaves as if the message Finish was received (enabling ADSE counts, deleting recorded identifiers…).

Alternatively, if the protocol is not a DC Rebalance protocol, Sink CPCM Instances with a recorded identifier will try to track the status of their peer Instance. If it has changed (CPCM Instance is an AD member for an AD Join protocol, is Blank for an AD Leave protocol, is a DC for a DC Transfer or a DC Split protocol, and if it is no longer a DC for a DC Merge protocol) it knows that the protocol was completed and thus behaves as if the message Finish was received.

For the CL Move protocol, Sink CPCM Instances with a recorded identifier request will request the CL again using the Get CL protocol (see clause 6.3.5.2.1). If the Source Instance replies with error 'Unknown CL Identifier', this means that it has erased the recorded identifier and thus means that the protocol was completed. The Sink Instance then behaves as if the message Finish was received (enabling the CL, deleting recorded identifiers, ...).

6.3.3.4.2 Client or server implementations

In order to decrease the risk of problems, client and server implementations are recommended to be implemented as atomic operations, i.e. to ensure that the relevant change of states will happen, once a protocol message has been received.

EXAMPLE: Dealing with the AD Leave Finish in an atomic way allows the protocol to be wholly completed (i.e. erasing the recorded CIC identifier and enabling the new ADSE counts) even if there is a power interruption before the message is fully processed. When the power returns, the process will continue and the protocol will be totally completed.

A possible way to reach this goal is to record protocol messages as soon as they arrive together with the status of each associated operation. In this way, if any problem occurs during the processing of the message, the instance will be able to perform all expected state changes. However, there is no need to send the response message as there is no means to know whether the peer instance is connected or not and as the SAC key has probably expired, it can thus no longer be used.

The server side can be implemented so that several protocols can be run simultaneously. In this case, the played protocol needs to be stored together with the recorded CIC identifier. Management of intermediate counts is detailed in the following clause. It is however not possible to cease being a Domain Controller (if the protocol was an ADM protocol) through a DC Transfer, a DC Merge or an AD Leave, unless loosing the ability to finish the protocol.

The client side implementation may only run one ADM protocol at a time. The protocol needs to be finished (i.e. the recorded CIC identifier has been erased) before starting a new protocol. Any number of CL Move protocols may be run simultaneously, but only one per Content Licence.

6.3.3.4.3 Intermediate counts

ADM protocol descriptions involve the update of ADSE counts, and also the keeping of intermediate counts in order to be able to rollback or finish the protocol. Intermediate counts (e.g. final counts for an AD Join or initial counts for an AD Leave) are always defined so that it is not possible to get additional ADSE credits by interrupting the protocol. These are the counts to be used if any other AD protocol is started. If the same protocol resumes, original counts are used.

There are at least two ways to keep these intermediate values:

- As full intermediate values as suggested in the normative parts of this multi-part deliverable.
- As differences that needs to be applied to the actual ADSE values in order to activate these intermediate counts. In this case, the protocol identifier also needs to be stored.

While both are functionally equivalent when only one AD protocol is run, the case becomes different for a server Instance that is capable of running several AD protocols simultaneously. As there is no means for the server Instance to know when (in case of interruption) and in which order the protocols will be finished, this way of implementing is very convenient as it will always lead to the same results whatever the actual order is.

6.3.3.4.4 Limits

Protocols are designed to be able to recover if the two peer CPCM Instances are re-connected, i.e. to cope with temporary network or power problems. If the two peer Instances never reconnect, this may result in some limitations for the user such as the inability to run a new protocol. Hence, implementations are recommended to allow the user to erase temporary values in order to enable the running of new protocols. This erasure may result in ADSE counts or CL loss, depending upon the stage at which the protocol is stopped (the Instance does not know it with certainty). It is recommended to propose the erasure only when the user is blocked from performing a requested action, namely:

- On the client side, for an ADM protocol, to run the same protocol with another DC or a different protocol.
- On the client side, for the CL Move protocol, to enable the Move of another CL if the source Instance does not have the capacity to record another CIC identifier.
- On the server side, to enable the running of another protocol if the sink Instance does not have the capacity to record another CIC identifier.

- On the server side, for an ADM protocol, if the user wishes to Transfer or Merge the DC (or even to let it Leave the AD).

When the user deletes the recorded CIC identifier and associated intermediate values, the same mechanisms as for the recovery apply:

- If the erasure is on the server side but not on the client side, the client may restart the protocol and this will be accepted by the server as a standard request.

- If the erasure is on the client side, the client will probably run another protocol as soon as the erasure happens, which will result in a change of state (this will be detected by the server), leading to the deletion of the recorded CIC identifier.

6.3.4 CPCM Security Controls Protocols

6.3.4.1 SAC related Protocol

SAC establishment protocols include the following protocols:

- SAC establishment protocol, which includes 3 messages: CPCM_AKE_init_message, CPCM_AKE_commit_message and CPCM_AKE_confirm_message and allows two CPCM Instances to share two secret session keys, one for message encryption and the other for message authentication. These keys have a limited lifetime defined by the C&R regime. SAC establishment is a necessary step before messages requiring SAC protection can be exchanged.

- SAC renewal protocol, which includes 3 messages; CPCM_AKE_renew_message, CPCM_AKE_commit_renew_message and CPCM_AKE_confirm_message. This protocol can be used by any of the two CPCM Instances before the current SAC keys actually expire thus allowing the new keys to be shared. The difference with the previous protocol sits in the fact that CPCM Instances only exchange their CIC identifiers instead of their whole certificate chain. Thus, using this protocol implies that CPCM Instances retain key K_{perm}; else, SAC establishment protocol has to be used.

- SAC termination, which consists of one single message; CPCM_AKE_terminate message. It can be sent by any of the parties to terminate the SAC. SAC keys will be no more usable once this message has been exchanged and thus can be erased.

Usage of the different SAC protocols is further explained in clause 7.4.

Specific error messages for SAC establishment are the following:

- "Incorrect certificate signature", used if the signature of any of the certificate signature in the chain is invalid.

- "Invalid certificate", used if the syntax of any of the certificate in the chain is incorrect.

- "Invalid certificate chain", used if the ancestor of a certificate cannot be found in the chain.

- "Certificate has been revoked", used when the Instance certificate has been revoked and the SAC is established for purposes other than Content Management.

NOTE: If the purpose for establishing the SAC is unknown, SAC can nevertheless be established. Generic error (see clause 6.3.2.16) will be used is the SAC is attempted to be used for an unpermitted purpose.

- "Ancestor certificate revoked", used when any of the parent certificates in the chain have been revoked.

- "Certificate C&R regime mismatch", used when the received certificate was delivered by an untrusted C&R regime.

- "Incorrect trust check value", used when the verification of the received check value fails. This error message may be used in response to the second or the third messages of the SAC establishment or the SAC renewal protocols.

- "SAC is not in place", used when a SAC renewal or a SAC termination request is made while the SAC does not exist.

The first six error messages may be used when a certificate is received, i.e. in response to the first or the second messages of the SAC establishment protocol.

6.3.4.2 AD Secret Management Protocols

AD secret management protocols include the following protocols:

- Deliver AD secret protocol that is used in the course of the AD join protocol. The protocol runs between the security control parts of the CPCM Instances and is used for the actual delivery of the AD Secret.
- Erase AD secret protocol that is used in the course of the AD leave protocol. The protocol runs between the security control parts of the CPCM Instances and ensures that the AD Secret is actually deleted from the leaving CPCM Instance.

Any attempt to run an AD secret management protocol outside of the applicable ADM protocol will be refused; error 'Protocol context error' will be generated (see clause 6.3.2).

All these messages are authenticated as they manipulate sensitive data. In addition, the Deliver AD secret message is encrypted as it carries the AD secret. SAC protection is applied.

Specific error messages for AD Secret Management protocols are:

- 'CPCM Instance is already a member of an AD' if the Joining Instance is already an AD member.
- 'CPCM Instance is not a member of any AD' if the Leaving Instance is a Blank Instance.
- 'No AD Secret had been stored' if the Leaving Instance already erased or never recorded the AD secret.

6.3.5 CPCM System and Content Management Protocols

The CPCM System and Content Management (SYS) protocols are designed to provide a wide variety of information exchange functions ranging from simple status enquiries through management of content exchange to verification of authorised domain membership and acquisition of revocation lists. The protocol set consists of the following functions, specified in the following clauses:

- CPCM Discovery protocol.
- CPCM Instance status enquiry.
- CPCM Content Licence exchange.
- CPCM Content operation permission.
- CPCM Content Item status enquiry.
- CPCM Device proximity checks.
- CPCM secure time exchange.
- CPCM Geographic information exchange.
- CPCM Instance AD membership verification.
- CPCM Content Move operation.
- CPCM Revocation List acquisition.
- CPCM Content Discovery Protocols.

Relevant clauses of the CPCM System Specification document list the protocol-specific message codes for CPCM System and Content Management, and provide a complete list of error codes and their meanings as defined for use in the CPCM security control protocols. It should be noted that not all of those listed errors within the CPCM System Specification are valid for every protocol call; those that are valid for a specific call are listed with that protocol call definition.

6.3.5.1 CPCM Instance Status Protocol

This protocol is used when a CPCM Instance wishes to find out the status of another CPCM Instance, i.e. for example, whether or not it is AD aware, whether or not it has ADM capabilities, whether or not it is absolute time capable, whether or not it is geographically aware, its content handling capability and so on.

EXAMPLE: CPCM Instance A wishes to store a CPCM Content Item on the home network. It sends the CPCM status enquiry message to CPCM Instance B, to verify its:

a) content handling capability;

b) AD awareness; and

c) its ADM capabilities.

This is so that Instance A can establish whether the CPCM Content Item can be stored by Instance B and handle it in the appropriate manner as dictated by the usage information associated with the content; for example, to ensure that another CPCM Instance requesting the content, is in the allowed AD to receive the CPCM Content Item.

This protocol allows for getting roughly the same information as the CPCM discovery protocol but is targeted to one single CPCM Instance. The first message is unicast and not broadcast. It is also less ADM oriented.

The response embeds two elements, the ADM status, that only carries static ADM Information and the CPCM information that carries capabilities of the CPCM Instance and the ADID, if any.

Messages of this protocol do not carry any sensitive data and hence need not any protection.

6.3.5.2 CPCM Content Licence Exchange

The CPCM Content Licence (CL) exchange process can be performed in two different ways, namely in Content Licence pull or Content License push modes. These two modes are described below.

6.3.5.2.1 CPCM CL Pull Mode

The CPCM_get_CL message, or pull mode, is used when a CPCM Instance, which wishes to receive a Content item, requests the associated CPCM Content Licence for that Content item from the CPCM Instance which is advertising the availability of that Content item.

The content_handling_operation field is included in the request and in the response. In the request, it gives the intended actions while in the response; it gives the authorized ones amongst the requested actions. The source CPCM Instance may also check that the requested actions match the capability of the requesting Instance.

If the content is requested through one or more Processing Entities and is marked MLAD, VLAD, MLocal or VLocal, the sink_id field is included in the request. It permits the source CPCM Instance to enforce proximity end-to-end. CL request message needs not any protection.

The CPCM Content Licence is returned in a '*CPCM_get_CL_response_message*'. The CPCM Content Licence pull protocol is shown figuratively within the relevant clause of the CPCM System Specification. If the content is marked VGAD or MGAD, and the CL is SAC protected, and is sent to a Processing Entity, then the response message also includes the CIC identifier of the source CPCM Instance so that the final destination CPCM Instance may assess its proximity with the source CPCM Instance. If auxiliary data are associated with the CL, they are also included in the message. The protection of this message depends on the CL protection mode. If the CL is under the sole protection of the AD secret, message does not need any protection. Else it is both authenticated and protected using SAC protection.

NOTE: If the CL is marked Copy No More and the CL is requested for a copy, '*CPCM_get_CL_response_message*' will not be returned but instead a CL Move protocol will be launched (see clause 6.3.5.9).

If a new version is required of an existing Content Licence associated with a particular CPCM Content item, then a special variant of the Content Licence pull protocol is used, namely '*CPCM_get_new_CL_message*'. This method is used when a CPCM Instance wishes to obtain extended rights for the associated CPCM Content item (see clause 6.5.5). If specific USI are requested, they may be included in the request. This message is thus authenticated using SAC protection to guarantee the integrity of requested rights.

There are two specific error codes associated with the CL request message: 'Unknown CL identifier' if the requested Instance has no CL with the requested CLID and 'Content handling operation refused' if the USI does not allow the CL to be transferred to the requesting Instance.

The last error code also applies when a new Content Licence is requested. Two other error codes may be generated in this case: 'invalid original content licence' if the request is made on the base of a Content Licence which is not correct in syntax or because the signature is not verified. 'Grant of USI refused' is generated if no new Content Licence could be generated.

Three error codes are associated with the response message: 'Content handling Operation mismatch' if the authorized operations in the response were not requested; 'Content Licence Syntax Error' if the received Content Licence cannot be parsed or if Auxiliary Data are missing; 'Content Licence signature error' if the signature of the CL is not valid, if the CL was protected using the AD secret while the USI do not allow for it (see clause 6.5.6).

6.3.5.2.2 CPCM CL Push Mode

CPCM Content Licence push protocol allows for a CPCM Instance to send a Content Licence to another CPCM Instance in absence of any positive request from that CPCM Instance. The '*CPCM_put_CL_message'* push mode is used firstly to provide a CPCM Content Licence to a Sink Device in advance of CPCM Content transfer; and secondly, to force a change of the content encryption key.

If the CL requires SAC protection, the message is signed and encrypted using SAC session keys.

If the CL requires only AD protection, the message does not need any further protection as the CL is already protected using the AD Secret.

Confirmation of the receipt of the passed Content License is achieved with the '*CPCM_put_CL_response_message'*. This message does not need any protection.

The message includes the authorized actions that can be performed with the content.

Auxiliary data, if any, are always pushed together with the CL. If the content licence is proximity or geographically restricted and pushed under SAC protection through one or more Processing Entities, the message also includes the CIC identifier of the Source CPCM Instance so that the final destination may perform a proximity test with it.

Three error codes are associated with the put CL message: 'Inappropriate Content handling Operation' if the authorized operations do not match the capabilities of the Sink CPCM Instance; 'Content Licence Syntax Error' if the received Content Licence cannot be parsed or if Auxiliary Data are missing; 'Content Licence signature error' if the signature of the CL is not valid, if the CL was protected using the AD secret while the USI do not allow for it (see clause 6.5.6).

6.3.5.3 CPCM Content Operation Permission

The CPCM Content operation permission protocol is used in two cases:

- When SVCA content is not received directly from the Acquisition Point that Acquired the content. In this case, the Sink CPCM Instance needs to verify that the value of Simultaneous View Count is not exceeded and runs the protocol with the delivering Acquisition Point to that end.
- When an Instance wants to send or access Remotely proximity-restricted content whose RAR is set to post record. In this case, it is necessary to first check whether the content is still direct or live.

The CPCM Content operation permission protocol consists of two calls - the request for permission for a particular CPCM Content operation from the first CPCM instance, and the corresponding response from the Acquisition Point that acquired the content, either granting or denying permission for the operation when the content is still direct or live. Alternatively, when content is no longer direct or live, the Acquisition Point issues a new Content Licence, using the *CPCM_put_CL_message*, as its response. The request message is not protected. The response message is authenticated by the SAC as its integrity needs to be guaranteed.

The original Content Licence may be added as an optional element in the request message, in which case the whole message will benefit from the same protection as that required for the Content Licence: signed and encrypted if the Licence requires SAC protection, but no additional protection otherwise.

There are two error codes associated with the request message: 'Unknown CL identifier', which is used if the AP does not know the CLID for which permission is requested, and 'invalid permission type' if the requested permission is not recognized.

6.3.5.4 CPCM Content Item Status Protocol

This protocol is used when a CPCM Instance wants to know whether or not a CPCM Content Item is currently being used, or handled, by another CPM Instance. It may be used by an Acquisition Point delivering count marked Simultaneous View Count Asserted (SVCA). Thanks to the protocol, if the upper limit of simultaneous views has already been reached, the AP may check whether all the views are still active in which case any new request for viewing will be denied or not, in which case a new request may be granted.

EXAMPLE 1: CPCM Instance A wishes to use a CPCM Content Item, which is marked SVCA. When obtaining the Content Licence, Instance A sees the content is marked SVCA and runs Content Operation Permission protocol with the Acquisition Point delivering the Content Item. The Acquisition Point has already reached the maximum number of simultaneous view counts. In this case, Instance A may directly be denied access to the content. However, CPCM Content Item Status protocol enables the Acquisition Point to check whether all current permissions granted to Consumption or Export Points are still active in order to determine whether a permission can be transferred. If one Instance replies that it has stopped displaying the Content, the Acquisition Point can grant the permission to CPCM Instance A to display the Content Item.

Included in the enquiry message will be the Content License ID for the Content Item being considered. The Enquiry message is sent to all CPCM Instances to which a Content Licence was delivered for that content. This Content License ID will be repeated in the response message. This allows matching the reply with the request as well as information about what the called on CPCM Instance is doing with the CPCM Content Item. The latter information is useful for those CPCM Instances, which are implementing several functional entities, to determine whether the CPCM Instance is actually viewing the Content or performing any other operation.

EXAMPLE 2: A PVR has requested the Content Licence to both record the content and display it. At some point in time, Rebecca leaves the living room where the PVR is to go to her bedroom and wishes to continue watching the Content from there. The recording is still on but now the content is no more displayed from the PVR. When the TV-set in the bedroom requests the permission to display the content, since the content is limited to one single simultaneous view, the Acquisition Point runs the Content Item Status Protocol with the PVR. The PVR replies it is still recording the content but no more displaying it. The Acquisition Point knows then that it can grant the permission to the TV-set.

The response message may also be used by a CP or an EP that stops consuming the content to proactively warn the AP of this status change.

Messages of this protocol do not require protection.

There is one error code associated with this protocol 'Unknown Content License ID'. This means either that the CPCM Instance has never dealt with a Content Item having such a CLID or it has stopped handling it.

NOTE: If the CPCM Instance receives a Content Item status request for a Content Item it is no more handling, it may either reply with an error or using the CPCM Item status response message with all bits set to zero, indicating that no action is currently performed on the content. The latter is preferable but it requires the implementation to keep track of past CLIDs.

6.3.5.5 Proximity tools

6.3.5.5.1 Introduction

Many CPCM operations or controls require for a CPCM Instance to assess whether another entity, be it a CPCM Instance or not, is Local to it or not. To that end, the CPCM specification provides a set of proximity tools. Unless otherwise specified by a CPCM C&R regime, the positive result of one proximity tool makes testing with another tool unnecessary. To provide interoperability, the CPCM specification requires the implementation of two proximity tools:

- RTT tool for a CPCM Instance to assess whether it is Local to an entity that is not CPCM aware.

- SRTT tool for a CPCM Instance to assess whether it is Local to a CPCM Instance.

In addition, CPCM specification allows for the combination of some of these proximity tools: this ability is itself specified as a tool named Proximity Through Association (see clause 6.3.5.5.5).

Proximity testing with a non-compliant CPCM device will only be used when storing content on a bit bucket. If the content is proximity restricted, the CPCM Instance may actually verify that the device with which it is exchanging the Content is Local or not. This verification may however be made optional by a C&R regime as CPCM has no means to control on the ability for the non-compliant device to receive or send the content remotely. CPCM has however other mechanisms to eventually control whether a user tries to use a proximity-restricted content that would have been moved remotely. This is done by performing a proximity check with the CPCM Instance that issued the Content Licence the latest whose identity is given by Last_CL_Issuer field in the Content Licence (see clause 6.2.2.4).

Proximity testing between CPCM Instances is used in the following cases:

- Handling of proximity restricted content
- Authorized Domain Management

Proximity tools dedicated to a given HN technology are also defined in TS 102 825-9 [i.11]. The usage of these tools assumes that messages exchanged between two devices are continuously carried over that HN technology. However, this is difficult for a device to determine whether this condition is met. Actually, it might happen that two CPCM devices use an IP connection to communicate with each other while the network path between the two CPCM devices uses a different technology.

Hence, before mandating the use of such tools, the governing C&R regime has to consider the probability of such an event. In any case, it may choose to endorse these HN-specific proximity tools in addition to those that are independent of the HN technology.

A C&R regime is also at liberty to define a proximity test specifically to enable a special business case. This may include cases where requirement on physical proximity is specifically relaxed to allow certain elements of the system to work, as illustrated in the scenario Hosted CPCM Service in TR 102 825-11 [i.8]. Such a proximity test returns a result based on the logical association of specific CPCM Instances. In this case, the hosted and home-located CPCM Instances which are not physically close together.

6.3.5.5.2 RTT

RTT tool is the standard Internet ping and is performed between a CPCM Instance and a non-compliant device. It is always initiated by the CPCM Instance.

A CPCM C&R regime will define the following parameters for that tool:

- For each network type, the maximum allowed ping response time to pass the test, RTT_maximum_time.
- In case of successful test, the maximum time during which there is no need to re-perform a new test, i.e. the CPCM Instance will still consider the non-compliant device to be Local.

A CPCM Instance may run several unsuccessful RTT tests before concluding the tested device is actually remote. One successful test is enough to consider the tested device as Local.

NOTE: To provide a good user experience, implementers will have to define a maximal number of failed trials until concluding the tested device is actually not Local.

6.3.5.5.3 SRTT

The security level provided by RTT is very low so the CPCM specification defines a Secure RTT (SRTT) tool which provides better security. As an example, the RTT tool is sensitive to man-in-the-middle attacks as the testing device has absolutely no means to verify whether the response is actually received from the tested device. SRTT is protected against man-in-the-middle attacks.

A CPCM C&R regime defines SRTT parameters that are analogous to the ones for RTT:

- For each network type, the maximum allowed ping response time to pass the test, SRTT_maximum_time.

- In case of successful test, the maximum time during which there is no need to re-perform a new test, i.e. the CPCM Instance will still consider the non-compliant device to be Local.

SRTT protocol may only be run after a SAC has been established between the two involved CPCM Instances. It consists of four messages:

- The first two messages, SRTT_request and SRTT_response, consist of a simple exchange of two random values. If the testing Instance receives the response before SRTT_maximum_time has elapsed, the testing Instance will proceed to the next step of the protocol. Else, as for RTT, it may restart the protocol as many times as needed until the implementation determines that the tested CPCM Instance is actually Remote.

- The last two messages, SRTT_validation_request and SRTT_validation_response, are used to verify that the SRTT_response was actually received from the tested CPCM Instance thus avoiding the possibility of the man-in-the-middle attack. The SRTT_validation_request is sent as soon as a message SRTT_response has been received in time. The tested Instance replies with the two random values exchanged in the first two messages, the message being authenticated using the SAC key or the AD key. The testing Instance checks that the random values are correct and that the message is correctly protected. If so, SRTT is successful. Else, implementations might restart the protocol or conclude that CPCM Instances are Remote.

Implementation of the first exchange is very sensitive as unnecessary delays may cause a Local Instance to be systematically viewed as Remote:

- Testing device needs to measure the elapsed time between the request sending and the response receiving. This means it is probably better to measure that time at the network layer rather than at the application layer to avoid measuring time due to internal processing delays. This also means that time measuring at the network level will be governed by the applicable C&R regime rules and that communication of time information between the different layers may need to be secured.

- Tested device should reply as quickly as possible to any SRTT_request message. Here again, this also means that the reply should probably be prepared at the network layer and obey the applicable C&R regime rules. Another recommendation is to prepare several random values in advance to be in a position to reply to many requests coming from many different Instances without taking any time to generate new random values.

The testing Instance has to retain exchanged random values before sending any SRTT validation request and will keep them until a response is received. Tested Instances have similarly to retain these values until either a validation request, a new SRTT request or an error message is received from the same Instance. It is recommended that a CPCM Instance is able to reply to SRTT requests from multiple requesting Instances at any time.

The SAC key is generally used for authenticating the last message: it is used for content management operations between CPCM Instances that are not members of an AD or that belong to a different AD, it is used in the AD Join protocol (AD secret may not be used since the Joining Instance does not know the AD secret). SAC key may be systematically used but requires a SAC establishment. When no SAC has been established, AD secret may be used between CPCM Instances that are members of the same AD.

In addition, the C&R regime may define adjustment values to cope with interfaces whose performances would be much different from usual network performances. These adjustment values will be used by devices using that interface when testing proximity. Adjustment values aim at coping with heterogeneous environments.

Adjustment value may be negative (for faster connections) or positive.

EXAMPLES: We suppose hereafter that SRTT_maximum_time is 7 ms.

If CPCM Instance A is connected through an Ethernet cable, with no adjustment value, to a router which is itself connected to CPCM Instance B only through Bluetooth, with 2 ms adjustment value, SRTT validation_request will be sent if SRTT_response is received within 9 ms whether A is the testing or the tested Instance.

If CPCM Instance C is connected through a new networking technology which is much faster than Ethernet, with 3 ms adjustment value, to a router which is connected to CPCM Instance D through Ethernet and through Bluetooth, SRTT validation request will be sent if SRTT response is received within 4 ms if Instance D received the request through Ethernet and 6 ms if it was received through Bluetooth.

If CPCM Instance E and F are each connected by Ethernet through two different routers, maximum time will be 7 ms whatever the network technology used between the two routers.

The only specific error to SRTT request message is 'Tested CPCM Instance does not support SRTT: use PTA', which is used if the tested CPCM Instance does not implement SRTT, which may be the case for CPCM Instance connected through a dedicated interface (for instance, ISO/IEC 7816 [i.29] interface as described in clause 9.4.2). In this case, the CPCM Instance may try to test proximity with this Instance using PTA as described in clause 6.3.5.5.5.

Errors specific to the SRTT response and SRTT validation response messages are:

- 'request challenge mismatch' and 'response challenge mismatch' occur if one or both of the random values contained in the SRTT validation response message is different from the ones exchanged in the first two messages. If both are different, any of the two error messages may be used.
- 'SRTT too high: proximity test restarted' is used when the measured time was too high but a new attempt is about to be made.
- 'SRTT too high: proximity test aborted' is used when the measured time was too high but no more attempts will be made.

There are no specific errors to other protocol messages.

6.3.5.5.4 GTTP

GTTP tool implementation is optional. It may only be implemented by devices that are geographically aware, in which case its implementation is highly recommended as it will provide a better accuracy of proximity than the SRTT tool and thus deliver a better user experience.

The tool uses one parameter, GTTP_Distance, which sets forth the maximum distance between devices to be viewed as Local.

The testing device requests the geographical location information of the second device using the CPCM geographic location protocol (see clause 6.3.5.7.2) compares it with its own geographical location and determines the distance between the two CPCM Instances.

6.3.5.5.5 PTA

Proximity Through Association is the method that allows a CPCM Instance to derive the proximity between itself and another CPCM Instance based on the fact that both are Local to a common intermediary CPCM Instance.

Not all proximity tests can be used in combination with each other, as this might lead to circumvention. SRTT and RTT may not be combined with themselves or each other, since in this case an SRTT or RTT test can nearly always be run between the two Instances that need to assess proximity without going through an intermediary. The same stands for a GTTP tool combined with itself.

PTA tool implementation is optional but recommended. It is highly recommended when a CPCM Instance implements several proximity methods and essential where there are non-IP interfaces to the CPCM Instance, such as a smart card interface.

Protocol messages are authenticated using the SAC. Therefore, SAC establishment needs to be completed before the PTA tool can be used.

There are three ways to run PTA:

- A CPCM Instance first assesses its proximity to the Intermediary CPCM Instance and, if it is successful, sends the identifier of the intermediary Instance and the used proximity method to the CPCM Instance with which it wants to assess proximity. The latter Instance then assesses its proximity to the identified intermediary CPCM Instance, using a proximity method that can be combined with the one initially used. It then replies to the originating CPCM Instance with the result of the test.

- Another possibility is to test its proximity to the intermediary CPCM Instance and then to ask the intermediary CPCM Instance to test its proximity to the target CPCM Instance and to reply with the result.

- The final possibility is to directly ask the intermediary CPCM Instance to test its proximity to both the originating CPCM Instance and to the target CPCM Instance and to reply with a combined result.

The same protocol messages are used regardless of which way is used. In the first method, the protocol is run with the target CPCM Instance while for the second it is run with the intermediary CPCM Instance. In both cases, the request message identifies which proximity method was used for the first proximity test, so that the target CPCM Instance or the intermediary CPCM Instance will only use a proximity test that can be legitimately combined with it. With the third method, the protocol is run with the intermediary CPCM Instance but the request message does not identify any proximity test method, so that the intermediary CPCM Instance will know that it has to perform both tests, and can thus decide for itself which proximity tests to apply.

A CPCM Instance implementing only one proximity test in addition to PTA is recommended to initially try one the first two PTA ways. If the Instance implements several proximity tests, the third way provides the intermediary CPCM Instance with a greater choice in the combinations of proximity methods it can use. The intermediary Instance may try several proximity test methods with each CPCM Instance before responding to the protocol message. Implementations need therefore to allow for a reasonable delay while this process takes place.

EXAMPLE: A smart card wants to assess its proximity with a CPCM Instance that resides in another networked device in the home. To that end:

- It may test its proximity with its host device using the PTDC tool and then request the target CPCM Instance to test its proximity with the smart card's host.

- It may test its proximity with its host device using the PTDC tool and then request its host to test proximity with the target CPCM Instance.

- It may request its host to test proximity with both itself and the target CPCM Instance, and allow the host to decide which tools to use for each test.

Three specific error codes are defined for this protocol: if the requested CPCM Instance cannot reach the third CPCM Instance, it generates error 'CPCM Instance unreachable'. If the proximity test method specified in the message is unknown, the requested CPCM Instance cannot assess whether the method can be used in association with another proximity test method and thus generates an error 'unknown proximity test method'. Finally, if the identified proximity test method is known but the requested Instance does not implement any method that can be combined with this, it replies with the error 'proximity method cannot be associated'. When no proximity method is included in the request message but no combination of proximity methods can be run with the requesting and the target CPCM Instance, the error 'proximity method cannot be associated' is generated.

6.3.5.5.6 PAAAA

The PAAAA tool may be used as a fallback solution if all other proximity tests fail. Its implementation is optional and may require the implementation of additional tools required by the proximity AAA.

To use PAAAA, the testing CPCM Instance requests the proximity AAA to check that it is local to another CPCM Instance. Once the AAA has determined the result of the test, which might involve proprietary means, it replies to the requesting CPCM Instance, reinserting the tested CPCM Instance identifier. Both messages are authenticated with the SAC key and thus require that a SAC has already been established prior to starting the test.

Protocol messages are authenticated using the SAC. Therefore, SAC establishment needs to be completed before the PAAAA tool can be used.

Two specific error codes are defined for this protocol: if the proximity AAA does not know the tested CPCM Instance id, or cannot connect to it, it generates error 'CPCM Instance unreachable'. If the CPCM identifier in the response does not match the one in the request, an error 'CPCM Instance mismatch' is generated.

6.3.5.5.7 Proximity Through Direct Connection (PTDC)

CPCM specification allows tools to assess whether two devices are directly connected, i.e. through a direct link like in the following examples:

- A smart card inserted in its host.
- A USB token inserted in a PC.
- Two devices connected through a wire with low electric latency.

Such tools are necessarily interface dependant and thus defined in TS 102 825-9 [i.11].

CPCM specification also specifies that PTDC may be used in association with other proximity tools (see clause 6.3.5.5.5). This is valid whatever the physical interface for which PTDC is defined.

6.3.5.6 CPCM Secure Time Protocol

This protocol is used to obtain secure absolute time from a CPCM Instance, Instance B, by another CPCM Instance, Instance A, which does not implement secure time itself.

EXAMPLE: Instance A acquires a CPCM Content Item which has time constraints associated with it, e.g. viewing window, Instance A does not implement secure time and so, to maintain the integrity of the CPCM Content Item, it requests the secure absolute time from Instance B, which does implement secure time

The request is unicast. The CPCM Instance may use the CPCM Discovery Protocol to know which Instances are time capable or check with a particular Instance using the CPCM Instance Status protocol. It may also make the request to a random Instance such as the Domain Controller. However there is a risk that the Instance will not be absolute time aware.

The response message is authenticated using the SAC. Therefore, SAC establishment needs to be completed before the Secure time protocol can be run.

The error codes associated with the request message here are as follows: 'CPCM Instance is not secure absolute time aware' indicating that the called upon CPCM Instance, Instance B in our example, is not secure time aware. This will happen for instance if Instance A did not check beforehand that Instance B was time aware. The other possible error is 'Absolute time currently not available' indicating that the absolute time is currently unavailable, in which case the requesting Instance may re-try to get absolute time later.

6.3.5.7 CPCM Geographic Information Protocol

This protocol is used when a CPCM Instance which is not geographically aware, wishes to verify its current region from another CPCM Instance within the Local Environment.

EXAMPLE: CPCM Instance A is a simple storage device which is about to acquire CPCM Content Item X, which has a geographical AD associated with it (i.e. for example, the user may want to send a content item straight to a storage device and view it later). CPCM Instance A is not geographically aware and sends an enquiry message (for verification of the geographic region it is in) to another device (CPCM Instance B) on the home network; this device could be the home gateway for example. This needs to be done before CPCM Instance A can handle the content item. Thus CPCM Instance B is called upon to confirm whether or not CPCM Instance A is within the geographic region signalled by CPCM Content Item X.

There are three protocols allowing to share or assess its geographical location: CPCM Enquire Geographic Location Formats, CPCM Get Geographic Location and CPCM Affirm Geographic Location.

6.3.5.7.1 CPCM Enquire Geographic Location Formats Protocol

This protocol is used by a CPCM Instance, CPCM Instance A, to determine which location information formats are supported by another CPCM Instance, CPCM Instance B, in the Local Environment. This needs to be done before the geographic location itself can be requested, as the information needs to be passed in a format which the requesting instance understands.

If there are no common formats supported between the requesting instance and the called upon instance, then this scenario becomes similar to one in which there are no geographic aware devices on the home network. In this case, the requesting CPCM Instance will need to revert to MLAD or VLAD usage rule.

The response message will contain a list of the formats supported by the called upon CPCM Instance, which will be included as a series of 1-byte identifiers with each identifier relating to one of the formats supported. The response is authenticated with the SAC secret, which means that a SAC has to be established before this protocol can be used.

The response message is authenticated using the SAC. Therefore, SAC establishment needs to be completed before the protocol can be run.

There are three error codes associated with this protocol. 'CPCM is not geographic aware' will be returned if the called upon CPCM Instance is not geographically aware. 'Geographical location format mismatch' will be returned by the requesting Instance if it supports none of the geographical formats that are supported by the called CPCM Instance.

6.3.5.7.2 CPCM Get Geographic Location

Once it has established which location information formats are supported by the called upon CPCM Instance, CPCM Instance B, the requesting CPCM Instance, CPCM Instance A, uses this protocol to request the geographic location in the indicated format.

This protocol can also be used as the first step in the Geographic Protocol, and as it includes a requested format type, can use this protocol as a means to find out if its particular format is supported by the called upon instance in a more centralised way than the request for all the supported formats and crosschecking these with its own. In other words, the requesting instance may start with this protocol thus requesting a particular format type, rather than going through the list of all the other formats supported by the called upon instance.

There are three possible errors associated with the enquiry message: 'CPCM Instance is not geographic aware', 'Geographic Location Format not supported', in the case that the requesting instance may not have performed the CPCM Enquire Geographic Location Formats step, and 'Geographic Location Information currently unavailable'.

The response message is authenticated using the SAC. Therefore, SAC establishment needs to be completed before the protocol can be run.

There is one error specific to the response message which is used when the geographic format of the response does not match the requested one.

6.3.5.7.3 CPCM Affirm Geographic Location

This protocol is used by a CPCM Instance to verify, or affirm, from another CPCM Instance in the local environment whether or not it is in the geographic location indicated by a CPCM Content Item.

The enquiry will include the geographic location information which is being assessed.

There are three errors associated with this message: CPCM Instance is not geographic aware, Geographic Location Format not supported and Geographic Information is currently unavailable.

The response will include an affirmation code (8 bits) which will indicate a yes or a no to the enquiry. The codes are described in TS 102 825-4 [i.9].

Protocol messages are authenticated using the SAC. Therefore, SAC establishment needs to be completed before the protocol can be run.

6.3.5.8 AD membership challenge Protocol

This protocol is used when a CPCM Instance wishes to securely verify whether or not another CPCM Instance belongs to an AD as indicated by a CPCM Content Item which it is handling (i.e. to verify whether the second CPCM Instance is in the same AD as the CPCM Instance currently handling the AD-bound CPCM Content Item).

EXAMPLE: A user wishes to move some content, CPCM Content Item X, from its current location which is a storage device, CPCM Instance A, on the user's home network to a mobile device, CPCM Instance B. CPCM Content Item X is bound to an AD. The storage device, the originating device for the content, in this transaction, CPCM Instance A, will have to verify whether or not the mobile device, CPCM Instance B, is a member of the AD to which the content item is bound before it can be passed on. In this case the mobile device's ADID needs to be the same as the storage device, the originating CPCM Instance A, as the content is only permitted in one AD.

The challenge message will contain the ADID of the challenging CPCM Instance, along with a random value, held in the "challenge" field, which is a nonce and prevents the message response from replay attacks.

There are four possible errors associated with this message: 'CPCM Instance is not AD aware', 'CPCM Instance is not a Member of the Indicated AD', 'CPCM Instance is Not a Member of any AD' and 'The Indicated AD has been Revoked'.

The response message is sent back if the called upon CPCM Instance is in the same AD as the challenging CPCM Instance, and will include the same nonce. It will also contain a signature, computed from the AD secret, guaranteeing that the responding Instance is actually a member of that AD.

There is a possible error which is associated with this response message: 'Response Challenge Mismatch: AD Membership Not Confirmed'. This may occur if the nonce returned in the response message is not the same value as that of the challenge message, hence indicating that the ADs may not be aligned.

6.3.5.9 Content Licence Move Protocol

This protocol is used Move a Content Licence that cannot be copied (i.e. marked Copy No More and/or marked VPA) from one CPCM Instance to another. It is triggered by the usual Content Licence exchange protocols (see clause 6.3.5.2) and hence both in pull mode or in push mode. This is the reason why there is no need for CL Move invite for this protocol. The protocol is a transaction protocol to avoid the loss of the Content Licence in case of problems.

When this protocol is run, it is assumed that the content movement is allowed by the USI. This has to be checked before starting the protocol.

Message CL Move Begin is the only message to be unprotected. All other messages are authenticated using the SAC secret. Message CL Move Response is also encrypted as it carries the whole CL. Therefore, a SAC is required to be set up before running the protocol.

The CLID is included in messages CL Move Begin, CL Move Ready, CL Move Commit, CL Move Confirm, CL Move Finish and CL Move Request. It is also implicitly included in message CL Move Response that carries the whole CL. Having the CLID allows to protocol to be identified if several CL Moves are performed simultaneously.

In addition, CL Move Ready and CL Move Confirm messages carry out the functional roles of the sending CPCM Instance, to confirm that the source is actually an AP, a PE or a SE and to confirm that the sink is a SE, a PE or an EP. The CL Move Response message also includes auxiliary data if any are associated with the Content Licence.

Specific error message codes for this protocol are:

- 'Unknown Content Licence identifier', in response to the CL Move Begin message, if the protocol was initiated for a different CLID (e.g. if there was a 'Get Content Licence' for another CL).
- 'CLID mismatch', in response to all messages except the CL Move Begin, if the message carries a CLID different to the one expected.
- 'Source CPCM Instance is not an AP, a PE or a SE', in response to CL Move Begin, CL Move Ready and CL Move Commit message, if the Source CPCM Instance has not a relevant role within the protocol.
- 'Destination CPCM Instance is not a SE, a PE or an EP', in response to CL Move Begin, CL Move Ready and CL Move Commit message, if the Destination CPCM Instance has not a relevant role within the protocol.

6.3.5.10 CPCM Discovery

This protocol is used when a CPCM Instance, Instance A, wishes to discover the CPCM-related information of other CPCM Instances, Instances B and C, on the network or in the local environment. The information will also include the ADM status information, as described in TS 102 825-4 [i.9]. The process will lead to the discovery of available authorised domains which Instance A can join as a new member.

EXAMPLE 1: The user Rebecca buys a new laptop (CPCM capable) which, along with its normal functionality, she wishes to use in her home network to view content stored on the network. When Rebecca connects the new device to the network, it broadcasts a call to discover what other CPCM devices there are on the network. These other devices, or instances, will unicast their responses informing of their CPCM status, and ADM status; also included will be the ADID of any ADs of which the Instances are a member, thus allowing the new device to discover what ADs are available for it to join.

Alternatively the requesting CPCM Instance A can narrow the discovery process and include an ADID for an authorised domain it is particularly concerned with. This might be the case if the CPCM Instance wants to get specific updates on the AD (e.g. who is the Local Master?) or if it wants to join that AD, based on the fact it wants to render a given piece of content that is bound to that AD.

EXAMPLE 2: Rebecca's network has 2 ADs associated with it, "Rebecca's Home Network" which is a physical (i.e. geographically-constrained) one encompassing her home network and "Rebecca Travel" which is a logical AD including mobile devices. Rebecca's new laptop has now joined the AD "Rebecca's Home Network", as a result of the requirements placed on it so far i.e. transfer of content which has so far been bound to the home network AD. Rebecca is halfway through viewing CPCM Content Item X on another device; as she is due to travel soon she wishes to transfer the content item to her new laptop, which she will be taking with her on her travels, in order to carry on viewing it while she is away. CPCM Content Item X is bound to the AD "Rebecca Travel" and as such can only be exchanged between devices belonging to this AD. In this case, upon initiating the transfer of content, the laptop, Instance A, discovers that it is not a member of the AD binding the content item and so broadcasts a call to other CPCM Instances on the network to find out whether they are a member of the relevant AD, (identified by including the ADID in the call) and what their membership status is i.e. are they simply members or can they do more? Instance B unicasts its response and informs Instance A that it is a member of the AD "Rebecca Travel" and that it has no ADM capabilities. Another instance, Instance C, also unicasts its response and informs Instance A that it is a member of the AD "Rebecca Travel" and that it has Local Master capabilities. From here, Instance A can join the required AD via Instance C and acquire the content.

NOTE: The requesting CPCM Instance may be performing the action upon delegation from another CPCM instance, in which case the instance id of the delegating CPCM Instance will be included in the message. The circumstances surrounding delegating CPCM Instances are better described in TS 102 825-4 [i.9].

The response messages are unicast from each responding CPCM Instance, and each message can have a variable payload length, The DC_ADSE_values and remotely_joined_cicf_list fields will only appear if the responding instance is a Domain Controller. The ADID field will only appear if the responding instance is an AD member and AD aware. delegating_instance_id is also transmitted when the message is delegated. Finally, the transmission of the AD name is optional. It is recommended to send it when the initial request was broadcast as the requester may not know the AD name and may need it for user interface purposes. If the initial request already embeds an ADID, embedding the AD name is less important as the requester is likely to already know it.

Finally, this protocol is very similar to the CPCM Instance status protocol. The main differences are that the initial message is broadcast and not unicast and that the response message allows the carrying of more ADM relative information.

Messages of this protocol do not carry any sensitive data and hence need not any protection.

There is one specific error message for the Discovery Response call which will be used by a CPCM Instance that wants to Join the AD while no LM or DC is present. This message will be used if any LM capable Instance is discovered. The error message will be sent to one, (or possibly all, LM capable Instances) so that a new LM may be elected.

6.3.5.11 Revocation List Acquisition Protocol

This protocol is used by a CPCM Instance to get updated Revocation Lists.

In order to be able to handle CPCM Content, a CPCM Instance should have a valid Revocation List as required in the Content Licence. If it does not currently have a Revocation List, then it will employ the CPCM Revocation List acquisition protocol to find out whether another CPCM Instance has a valid list. The Revocation List Transfer itself is dependent upon the underlying network technology. There are two protocol messages used; Get CPCM Revocation List and Notify CPCM Revocation List.

The requesting CPCM Instance inserts a C&R regime mask into the requesting message. In this C&R regime mask, all the bits of the requested revocation lists are set and, for each of the bit set, the requested RL_index is indicated.

Requesting message can be broadcast or sent:

- If the requesting Instance does not know whether the RL it needs are available, the message is broadcast. Any CPCM Instance that knows at least one of the requested RL will reply with all the requested RL it knows.
- If the requesting Instance knows a given CPCM Instance has the RL it needs (e.g. through the CPCM discovery protocol), it sends the message to that Instance rather than broadcasting it.

A CPCM Instance receiving a RL Enquiry does not need to have all the requested Revocation Lists to reply to the message. It replies always if it knows only some of them. It replies using one message for each of the Revocation List it can provide. Each message embeds an RL_locator which provides relevant information for the HN layer to actually download the RL. If it does not have any of the requested revocation lists, it replies with an error message, 'Revocation List not available'.

Messages of this protocol do not carry any sensitive data, since the Revocation List is signed, and hence do not need any protection.

If a requesting CPCM Instance receives a RL it did not request, it may signal it to the sending CPCM Instance using error 'Revocation List Mismatch'. In this case, that Instance may, but is not required to, re-send the requested RL.

6.3.5.12 CPCM Content Discovery Protocol

A CPCM Instance uses this protocol to discover from another CPCM Instance, the usage rights and authorised domain associated with a particular CPCM Content Item. A CPCM instance will normally perform this operation after it has discovered the content item using the HN technology specific protocol, as described in TS 102 825-9 [i.11], or after the user has initiated exchange of a particular content item.

The protocol consists of two steps: the Content Discovery Request message and the Content Discovery Response message. The response embeds two elements, the ADM status, that only carries static ADM Information and the CPCM information that carries capabilities of the CPCM Instance and the ADID, if any.

Messages of this protocol do not carry any sensitive data and hence need not any protection.

There is one error code associated with this message which returns a value expressing that the Content Licence ID is unknown. This could occur, for example, if the called upon CPCM Instance is not aware of the content item in question.

6.3.6 CPCM AD Management Protocols

CPCM has a specific set of fields and message protocols that are only used for ADM management. All these protocols and enumerated values can be ignored by CPCM implementations that are not AD aware.

Some of the ADM protocols may use the delegation mechanism that allows a Local Master to act as a proxy between a Domain Controller and another CPCM Instance. The CPCM specification defines which messages can be delegated. To be future proof, it also reserves code values for future messages that would be delegatable. It is recommended that LM capable implementations of this version of the specification should delegate all messages with codes in this range when they are LM.

6.3.6.1 ADM Enumerated fields

As mentioned in TS 102 825-4 [i.9], the ADM Protocols make use of some additional enumerated fields which are specific to AD Management.

6.3.6.1.1 ADM Status

This field gives the status of a CPCM Instance with regard to ADM i.e. it describes the capability of the CPCM Instance in terms of whether it is a DC, LM, both, a simple AD member or a Blank Instance.

6.3.6.1.2 ADM Condition

This field indicates the reason for which intervention from the ADMAAA is necessary. It is only carried when using the ADMAAA tool.

6.3.6.1.3 ADM Protocol

This field is carried when there is an invitation to start an AD protocol (see clause 6.3.6.2.4) and describes which protocol is to be started, i.e. AD Join, AD Leaving, DC Transfer, Merge, Split or Rebalance.

6.3.6.1.4 Delegating Instance Identifier

This field indicates that a delegated action is being performed and carries either the CIC identifier of the requesting CPCM Instance in a request message or the CIC identifier of the DC to which the action was delegated in a response message.

6.3.6.1.5 Local Master Capability

This field is carried in the election of a Local Master. It describes the LM capability of a particular CPCM Instance. Three states correspond to the CPCM Instance capabilities or status; LM capable, DC capable or current DC, and two of them can be used under special circumstances to avoid becoming Local Master (e.g. because the CPCM Instance is about to shut down,) or to be the next Local Master (e.g. for a DC that is about to proceed to a DC Transfer).

6.3.6.1.6 Remotely Joined CICF list

This field carries the list of CICF that Joined Remotely and allows management of corresponding CPCM Instances. It is used if such an Instance becomes Local to the Domain Controller so that the original Join can be re-counted as Local if other ADSE conditions are met or upon Leaving so that it is counted as a Remote Leave.

6.3.6.2 General ADM Protocols

6.3.6.2.1 AD update protocol

This protocol is used by each reconnecting CPCM Instance to get the latest information about the AD. The requesting CPCM Instance sends to the Local Master its current version of the AD Internal record list. The Local Master compares this list with its own one. If the Local Master has at least one AD internal record that is more recent (i.e. with a higher index), it sends back a list including all the more recent AD internal records. If the requesting CPCM Instance has at least one more recent AD internal record, it updates all the corresponding records and broadcasts a message AD update Indication so that all the connected CPCM Instances from the AD are informed.

Local Masters and Domain Controllers also broadcast a message AD update indication as soon as an AD internal record is updated.

All protocol messages are authenticated by the AD secret to guarantee the integrity of the exchanged AD internal records.

The protocol has three messages; AD update request, AD update response, AD update indication. Two errors can occur with all the three messages: 'Internal record list is inconsistent' if any inconsistency has been detected in the transmitted list. An inconsistency can for instance be a list of internal records corresponding to distinct ADs; 'Internal record with bad signature' if the signature of at least one record is not correct.

In addition, in response to message AD update request, error "Instance is neither a Local Master nor a Domain Controller" may also be generated if the message is received by a simple AD member.

The following two errors may be generated for both AD update request and AD update indication: 'Receiving Instance is not a member of the indicated AD' if the receiving Instance does not belong to the same AD as the internal record list one. 'The indicated AD has been revoked' if the AD is revoked.

In response to messages AD update response and AD update indication, error 'Outdated internal record list' may be generated if the message was embedding an internal record list including at least one record that is less recent than the one recorded by the destination CPCM Instance.

Another error can be generated for the AD update response message: 'ADID mismatch' if the received list corresponds to an AD other than the one of the AD update request message.

One additional error may also be generated in response to message AD update indication: 'Receiving Instance is not AD aware' if the receiving Instance is not AD aware.

NOTE: Error 'Receiving Instance is not ADM capable' is not expected to be generated as this Instance will behave as any other Instance from the AD that does not record the AD internal record.

6.3.6.2.2 AD change protocol

This protocol is used when the user wants to change the human-readable name of their AD, e.g. if this one was allocated by default at the AD creation. The change is performed at a Domain Controller. The message is delegatable, thus allowing an Instance to send the change request to the Local Master that forwards the request to a Domain Controller.

Protocol messages carry the ADID and the new AD name.

When the name has been changed, a new internal record is generated. If the AD has several DCs, other DCs will detect the name change when receiving the updated internal record, and generate a new internal record reflecting the name change and other necessary changes (new index, generation date if present, etc.).

Protocol messages are authenticated using the AD secret to ensure the request is coming from a member of the AD.

The following error may be generated for both protocol messages: 'ADID mismatch' if the message does not carry the relevant ADID. In addition, the following errors may be generated in response to message AD change request: 'No Domain Controller available' if the message was sent to a Local Master that could not contact any Domain Controller; 'Instance is neither a Domain controller nor a Local Master' if the request was sent to a simple AD member. 'Receiving Instance is not a Domain Controller' if the message was delegated but not to a Domain Controller; 'Renaming refused' if the Domain controller refused to proceed to the renaming, e.g. because the AD is locked; 'The indicated AD has been revoked' if the AD was revoked. In addition, error 'name mismatch' can be generated in response to message AD update response if the new name is different from the requested one; 'Sending Instance is neither a Domain Controller nor a Local Master' if the response does not come from a Local Master or a Domain Controller; 'Sending Instance is not a Domain Controller' if the message was delegated but not to a Domain Controller.

6.3.6.2.3 AD quorum test protocol

This protocol is used when a quorum test is needed during the ADSE enforcement (see clause 8.2.7). The request message is sent by the requesting Domain Controller to all connected CICFs. The reply allows the DC to check whether the CICF is actually connected and is from the same AD. It will then perform a proximity test with each of the replying Instances to determine which ones are in the Local Environment. It will then be able to determine whether the quorum is reached or not.

The request carries the ADID and a random value that is changed at each new quorum test. The reply is signed using the AD secret, thus validating the AD membership of the replying Instance. It includes the CIC identifier of the replying Instance to distinguish between responses from two different Instances, and the AD capability to check the Instance is a CICF.

Error 'The indicated AD has been revoked' will be generated for the query message if the AD was revoked, 'Receiving Instance is not member of the indicated AD' will be generated for the query message if receiving Instance belongs to another AD. Error 'challenge mismatch in quorum test response' is generated if the challenge embedded in the response message is not the one that was sent. Finally, error 'Receiving Instance is not ADSE countable' is generated for the query message if the replying Instance is not a CICF and error 'Sending Instance is not ADSE countable' is generated for the response message in the same case.

6.3.6.2.4 ADM invite protocol

This protocol is used when the user wants a CPCM Instance on another device to perform an ADM protocol (Join, Leave, DC Transfer, Merge, Split or Rebalance). The message ADM Invite is sent to the Instance that needs to perform the protocol as a client. Results of the ADM protocol are then carried in the corresponding response message.

The protocol is also used to resume an ADM protocol that was interrupted (see clause 6.3.3.4.1).

The request carries the identifier of the ADM protocol that is to be run and the ADID of the applicable AD. In the case of an invitation to Join, the invitation may carry the ADID of the AD to Join, or it might be set to zero if no particular AD is requested.

The request message may optionally embed a DC identifier if the user expects the protocol to be run with a particular DC. Else, the invited Instance may choose any DC or leave the choice to the Local Master, if applicable. It may also embed some ADSE values in the case of a DC Split or a DC rebalance. The values indicate then how the split or the rebalance is expected to be done.

The response message identifies the protocol that was run, and the ADID of the corresponding ADM.

The ADM Invite Protocol messages do not carry any sensitive information and hence are not protected.

NOTE: There is a small risk of a faulty or malicious device generating excessive ADM Invite protocol messages. It is not practical to sign these messages using the SAC or the AD secret as an invite may come from a non-AD member or a non-CPCM Instance. Therefore, implementers may wish to ignore such messages should they arrive in suspicious quantity or in a suspicious context. In this case, it is recommended to display corresponding information to the user, provided that this in itself does not further disrupt the use of the system.

The following error messages may be generated in response to the request message:

- 'no Domain Controller is available' if the invited Instance could not contact any Domain Controller, including through a Local Master.
- 'Requested Domain Controller is not available' if a specific DC was requested and it could not be contacted, including through a Local Master.
- 'Requested Instance is not a Domain Controller' if a specific DC was requested but the corresponding CPCM Instance is not a DC.
- 'Receiving Instance is not AD aware' in case of a Join and the invited Instance is not AD aware.
- 'Receiving Instance is not ADM capable' in case of a Join and the invited Instance is not ADM capable.
- 'History_ceiling exceeded' if the protocol is a Join but the invited Instance cannot Join any more ADs.
- 'Receiving Instance is not member of any AD' if the Instance is Blank and the requested protocol is not a Join.
- 'Receiving Instance is not member of the indicated AD' if an ADID is specified in the message but the receiving Instance is a member of a different AD.
- 'Source Instance is already AD member' if the protocol is a Join but the receiving Instance is already member of an AD.
- 'Receiving Instance is not Domain Controller capable' if the requested protocol is a DC Transfer or a DC Split and the invited Instance is not DC capable.
- 'Receiving Instance is not a Domain Controller' if the protocol is a Rebalance or a Merge but the invited Instance is not a Domain Controller.
- 'Protocol cannot be resumed' if the protocol status flag is asserted but the CPCM Instance is not currently running the ADM protocol with the requesting CPCM Instance.

Error messages explaining the protocol failure may also be forwarded in reply to the request message.

The following error messages may be generated in reply to the response message:

- 'ADID mismatch' if a specific ADID was requested but the response message carries a different ADID.
- 'Protocol mismatch' is the response mismatch carries a different protocol to the requested one.

6.3.6.3 Local Master Election

This protocol can be used to elect a new Local Master as soon as it is detected that the previous one is no longer connected. It may also be run in other situations. For instance, if an AD member wants to become a Domain Controller, it will launch a new LM Election to become the Local Master. This avoids unnecessary use of the delegation mechanism.

All the messages are broadcast and protected by the AD Secret. This avoids easy "denial of service" attacks done by devices that are not AD members, or even that are not AD aware, that would repeatedly launch new LM Elections. If a LM Election is needed by a non-AD member, protocol described in clause 8.3.2.7 can be used.

If the CPCM Instance requesting the LM Election is LM capable, it broadcasts its LM capability. A non LM capable CPCM Instance may also launch the protocol (e.g. if it needs a Local Master for an AD operation) as long as it is a member of the AD. This is the only case where the LM_capability field may be absent from the message. Each CPCM Instance with higher capability replies with its own capability. If the requesting or replying CPCM Instance is a Domain Controller, it also includes its current ADSE counts. Further details are given in clause 8.3.2.7.

Protocol messages also carry the ADID and the status of the Instance in order to allow for consistency checks.

The LM_election_indication_message is broadcast by the elected LM, to inform all other Instances on the network of its new status.

Error codes associated with this protocol are:

- 'The indicated AD has been revoked', that can be sent by any AD member, including non-LM capable members, in response to an LM Election request or LM Election Indication if it detects that the LM Election concerns a revoked AD (see also clause 7.8.2.4).
- 'Receiving Instance is not member of the indicated AD' when a CPCM Instance receives a LM Election Request or LM Election Indication message from another AD.
- 'ADID mismatch' when the response to a LM Election Request does not carry the expected ADID.
- 'Receiving Instance is not LM capable' if a LM Election Request is sent to an Instance that is not LM capable.
- 'Inconsistent ADM status' if there is a mismatch between the status and the LM capability indicated in a LM Election Request or Response message.

  EXAMPLE: status indicates the instance is DC while the LM_capability says the device is only LM capable.

- 'LM capability too low' if the response carries a LM capability that is lower than the one of the request or if the replying Instance is not LM capable. This response is thus ignored.
- Too many Local Masters' sent by any CPCM Instance receiving several consecutive LM Election Indications for the same AD. In this case, it is recommended that a new LM election is launched by one of the elected LMs.

6.3.6.4 ADM Transaction protocols

6.3.6.4.1 Common elements

ADM Transaction protocols share a common design and therefore have commonalities regarding elements that are carried in the different messages, regarding message protection or regarding errors that may be generated with respect to the different messages. This clause addresses these commonalities. The following clauses tackle specificities for each of the protocols.

Begin message is the only one to be unprotected. It is actually assumed that no SAC is in place when the protocol starts. Even if a SAC is in place, transaction protocols require the SAC to be renewed, and there is thus no reason to use a different SAC Secret for the first protocol message.

All other messages are at least SAC protected. In addition, two messages are AD protected in order to confirm that both parties actually know the AD Secret. The two messages are different from one protocol to another. This is due to the fact that for AD Join and AD Leave protocols, the AD Secret is enabled at different steps in the protocol.

Unless otherwise stated, all messages are unicast.

The Begin, Ready and Commit messages always carry the status of the CPCM Instance and the ADID. This allows for verifying that both CPCM Instances have the correct status to start the protocol and are in the same AD (for the Join protocol, ADID is only set if a specific AD is requested in the Begin message). ADID and status are repeated in the Commit message because the Begin message is not protected: this allows the values to be confirmed.

The Finish message never carries any element, excepted in the case of delegation (see clause 8.6.4).

Error codes associated with Begin messages are:

- 'Receiving Instance is not a Domain Controller' if the Begin message was sent to an Instance that is not a Domain Controller. For Join and Leave protocols, this error may only be generated if the message was delegated to an Instance that is not a Domain Controller.
- 'Receiving Instance is not a member of any AD' if the message was sent to a Blank Instance.
- 'Receiving Instance is not a member of the indicated AD' if the message was sent to an Instance that is a member of a different AD. For the Join protocol, it is generated if a specific ADID was requested, but the replying Instance is not a member of that AD.
- 'Receiving Instance is not ADM capable' if the message was sent to a CPCM Instance that cannot run AD protocols.
- 'Sending Instance is not ADM capable' if the message was sent by a CPCM Instance that cannot run AD protocols.
- 'The indicated AD has been revoked' if the AD is revoked. For a Join message, a specific ADID was requested but the corresponding AD was revoked. This cannot happen when no specific ADID was requested.
- 'Certificate message mismatch' if the AD capability indicated in the message does not match the one indicated in the certificate (which is exchanged during SAC establishment).

Error codes associated with Ready messages are:

- 'Certificate message mismatch' if the AD capability indicated in the certificate shows that the Instance is not Domain Controller capable and hence the CPCM Instance was not expected to answer. The same error is used for non-CICF Instances in the Join or Leave protocol when the replying Instance is not LM capable.
- 'ADID mismatch' if the ADID in the message does not match the one of the Begin message. This cannot happen for Join protocols when no specific ADID was requested.

'ADID mismatch' may also be generated in response to Commit messages if the ADID is different from the one in the Begin or in the Ready message. In addition, error 'inconsistent ADM status' may be generated if the sent status is not the same as the one in the Begin message.

There is no specific error codes associated with Finish messages.

6.3.6.4.2 AD Join

This protocol is used to when a Blank Instance joins an AD. The protocol is a transaction protocol as described in clause 6.3.2.15 and hence is composed of five messages. The protocol can be delegated, in which case the delegating_instance_id is carried in the messages as described in clause 8.6.4.

Messages AD Join Confirm and AD Join Finish are AD protected. This allows for confirming that the AD Secret was properly transmitted, and thus confirming AD membership. If AD Secret authentication fails at that step, the ADM control of the DC may request its Security Control to deliver the AD Secret again and to resume the protocol at that step.

The AD Join begin message is normally unicast but may be broadcast by non-self-managing instances (see clause 8.2.5) in which case it has a different message identifier 0x600C instead of 0x600B. All other messages are unicast.

In addition to elements specified in 6.3.6.4.1, the AD Join Begin carries the ADM capability in order for the receiving CPCM Instance to be able to check that the Instance is ADM capable. It also allows a Local Master to know whether the message needs to be delegated or not (i.e. if the Instance is ADSE countable). This capability is checked upon SAC establishment and thus does not need to be repeated in the AD Join commit message.

ADID in the AD Join Begin message can be set to zero if no specific ADID was requested. However, in the Commit message, it has to be set and shows that the ADID was properly recorded by the Joining Instance.

AD Join confirm message does not carry any CPCM elements excepted in the case of delegation (see clause 8.6.4).

In addition to the ones given in clause 6.3.6.4.1, error codes associated with AD Join Begin are:

- 'Remote ceiling exceeded' if the Join is Remote and Remote ceiling has already been reached.
- 'Local ceiling exceeded ad negative quorum test' of the Join is Remote and both local ceiling has been reached and quorum test is negative.
- 'Total ceiling exceeded' if the total ceiling has been exceeded.
- 'Rejection by ADMAAA' if there was an attempt to run ADM AAA but this was rejected.
- 'No Domain Controller is available' which is used when the message was addressed to a LM that cannot contact any DC.

'Requested Domain Controller is not available' if the Join was requested with a specific DC (e.g. following an ADM invite') but the latter is not available:

- 'Requested Instance is not a Domain Controller' available' if the Join was requested with a specific DC but the corresponding CPCM Instance is not Domain Controller.
- 'Receiving Instance is neither a Domain Controller nor a Local Master' if the Join request was sent to a simple AD member.
- 'Source Instance is already AD member' if the requesting Instance is already an AD member.

Error codes associated with AD Join Ready are:

- 'Sending Instance is not Domain Controller' which is used if the requesting Instance is a CICF and the protocol is not run with a DC. This is also used if the message was delegated but not to a Domain Controller.
- 'Sending Instance is neither a Domain Controller nor a Local Master' which is used if the response does not come from a Local Master or a Domain Controller.
- 'The indicated AD has been revoked' if the AD proposed for Joining is revoked. This should not happen if a specific ADID was requested, as the Source CPCM Instance would have known that the AD is revoked.

Other messages have no additional specific error codes.

6.3.6.4.3 AD Leave

The AD Leave protocol is used when an Instance is to be removed from the AD. It is designed to ensure that the ADSE counts that were consumed when the Instance originally Joined are recovered for future use, and to ensure that the Instance no longer has access to AD-bound content. The AD Leave protocol is run between the leaving Instance and a DC. If the Instance is not aware of the DC, it can perform a delegated AD Leave via the Local Master. If the Instance is non-CICF, it can be run directly with an LM without delegation to a DC.

NOTE: In theory, a non-CICF, (i.e. one that does not include Consumption or Export Point functionality), can Leave the domain simply by deleting the AD records and secrets (e.g. a reset to factory settings). However, using the AD Leave protocol requires confirmation, and thus provides additional confidence that this is what the user truly desires. Implementations receiving an ADM Invite (Leave) message should especially avoid such a simple approach to reduce risk of denial of service attacks based on this protocol.

AD Leave Ready and AD Leave Commit are AD protected to ensure that both Instances actually belong to the same AD.

The other protocol messages do not carry any CPCM elements. Protocol calls do not carry any additional elements to the one given in clause 6.3.6.4.1.

In addition to the ones given in clause 6.3.6.4.1, error codes associated with AD Leave Begin are:

- 'No Domain Controller is available' which is used when the message was addressed to a LM that cannot contact any DC.
- 'Requested Domain Controller is not available' if the Leave was requested with a specific DC (e.g. following an ADM Invite (Leave)) but the latter is not available.
- 'Requested Instance is not a Domain Controller' if the Leave was requested with a specific DC (e.g. following an ADM Invite (Leave)) but the latter is not currently configured as a DC.
- 'Receiving Instance is neither a Domain Controller nor a Local Master' if the Leave request was sent to a simple AD member.
- 'Sending CPCM Instance is not a member of any AD' which is used when the Instance issuing the AD Leave Begin is a Blank Instance.

Error codes associated with AD Leave Ready are:

- 'Sending Instance is not a Domain Controller' which is used if the requesting Instance is a CICF and the protocol is not run with a DC. This is also used if the message was delegated but not to a Domain Controller.
- 'Sending Instance is neither a Domain Controller nor a Local Master' which is used if the response does not come from a Local Master or a Domain Controller.

Other messages have no additional specific error codes.

6.3.6.4.4 DC Transfer

This protocol is used to when the user wishes to move the Domain Controller function to a different device. The protocol is a transaction protocol as described in clause 6.3.2.16 and hence is composed of five messages. The protocol is run directly between the "new" DC and an "old" DC, and cannot be delegated. The new DC has to be a DC-capable CPCM Instance which is a member of the AD but is not currently serving as a DC.

NOTE: Transferring a DC function has no effect on any other DCs that may be present in the AD. The transfer simply moves one of the DC functions between devices. The total number of DCs in the AD will therefore remain the same.

DC Transfer Ready and Commit messages are signed with the AD Secret to verify that both Instances actually know the AD Secret.

In addition to elements given in clause 6.3.6.4.1, the DC Transfer Begin message carries the capability of the Source CPCM Instance in order to enable the current DC to verify it is DC capable. This capability is compared with the one of the certificate during the SAC establishment and hence the DC Transfer Commit does not need to carry it again.

The DC Transfer Confirm message carries the latest AD internal record of the previous DC. It is from this internal record that the new DC will generate a new AD internal record to inform other CPCM Instances of the DC Transfer.

The "Become new DC" message carries the current ADSE_counts and list of CICFs that Remotely Joined that will be used by the future DC.

The other protocol messages do not carry any additional CPCM elements.

In addition to error codes given in clause 6.3.6.4.1, error codes associated with DC Transfer Begin are:

- 'DC_remote_ceiling exceeded' can occur during remote operations when the C&R Regime defined limit is exceeded.
- 'Rejection by ADMAAA' occurs if the DC Transfer operation has exceeded the defined limits, and the fallback ADMAAA method does not permit these to be overridden.
- 'Sending Instance is not a member of any AD' if the Source Instance is a Blank Instance.
- 'Sending Instance is not Domain Controller capable' can occur when the capability indicates the CPCM Instance is not DC capable.

Error codes associated with DC Transfer Ready are:

- 'Sending Instance is not Domain Controller' if the message was not received from a DC. This should not normally occur.

Error codes associated with DC Transfer Confirm are:

- 'Internal_record with bad signature' if the received domain internal data record is corrupted.
- 'Inconsistent internal_record' if the received AD internal record is not consistent: for instance, this will be the case if it was not generated by the sink CPCM Instance.
- 'Outdated internal_record' if the received domain internal data record is known to be out of date (usually because the New DC already has a copy of a later record).
- 'ADID mismatch' if the received internal record corresponds to another AD.

Error codes for Become Domain Controller are:

- 'Unexpected ADSE values' if the current received values are not internally consistent (e.g. if total_count is not the sum of remote_count and local_count) or if any other problem is detected.
- 'Unexpected list of CICFs that Joined Remotely' if the received list seems inconsistent (number of elements is different from remote_count, identifier of the peer Instance is in the list).

Other messages have no additional specific error codes.

6.3.6.4.5 DC Split

This protocol is used when the user wants to have an additional Domain Controller, for instance to allow easy AD Joining in a second home. ADSE counts and possibly some CICFs identifiers of the list of CICFs that Remotely Joined are split and transferred from the Sink Instance, a current DC, to the Source Instance, the future additional DC.

This protocol does not use the delegation process as DCs are supposed to have the capacity to discover and communicate with each other directly.

DC Split Ready and DC Split Commit messages are also AD protected in order to confirm that both the splitting and the split DC actually know the AD Secret.

In addition to elements given in clause 6.3.6.4.1, the DC Split Begin message carries the AD capability of the Source Instance so that the Sink Domain Controller may check that the Instance is actually DC capable.

DC Split Ready message carries the current ADSE values and the list of CICFs that Joined Remotely in order for the future DC to be able to select the ADSE counts, ceilings and the CICF identifiers that are to be split.

The DC Split Commit message carries the ADSE values and the list of CICF identifiers that are to be transferred. Values are positive and need to be lower than or equal to the values received in the DC Split Ready message.

Finally, the DC Split Confirm message carries the future first internal record of the new DC. This allows the new DC to know for instance its DC_local_id. This internal record is then directly broadcast by the new DC.

In addition to codes given in clause 6.3.6.4.1, error codes associated with DC Split Begin are:

- 'DC_remote_ceiling exceeded' can occur during remote operations when the C&R Regime defined limit is reached.
- 'DC_split_ceiling exceeded' can occur if the maximum number of DCs in the AD is reached.
- 'Rejection by ADMAAA' occurs if the DC Split operation has exceeded the defined limits, and the fallback ADMAAA method does not permit these to be overridden.
- 'Sending Instance is not a member of any AD' if the Source Instance is a Blank Instance.
- 'Sending Instance is not Domain Controller capable' can occur when the capability indicates that the CPCM Instance is not DC capable.

Error codes associated with DC Split Ready are:

- 'Sending Instance is not a Domain Controller' if the replying CPCM Instance is not a DC. This may happen if the Begin message was not actually sent by that CPCM Instance.
- 'Unexpected ADSE values' if the current received values are not internally consistent (e.g. if total_count is not the sum of remote_count and local_count) or if any other problem is detected.
- 'Unexpected list of CICFs that Joined Remotely' if the received list seems inconsistent (number of elements is different from remote_count, identifier of the peer Instance is in the list).

Error codes associated with DC Split Commit are:

- 'Bad ADSE counts requested' if at least one requested ADSE count exceed the value of the Sink DC Controller or if the requested ADSE counts are not consistent or have higher values than the one sent in the previous message. In this case, The Sink DC controller has the possibility not to send any error but to propose consistent ADSE counts in the DC Rebalance message.
- 'Bad CICF identifier requested' if at least one of the requested CICF identifiers is not in the list of the Sink DC. In this case, The Sink DC controller has the possibility not to send any error but to directly remove the corresponding CICFs identifiers from the requested list.

Error codes associated with DC Split Confirm are:

- 'Internal_record with bad signature' if the received domain internal data record is corrupt.
- 'Inconsistent internal_record' if the received AD internal record is not consistent: For instant, this will be the case if it was not generated with its own CIC, with index 0 or with an already allocated DC_local_id.
- 'ADID mismatch' if the received internal record corresponds to another AD.

6.3.6.4.6 DC Merge

This protocol is used to when the user wants to merge two of its Domain Controllers, ADSE counts and possibily some CICF identifiers of the list of CICFs that Remotely Joined, are merged and transferred from the Sink Instance,the merging DC, future simple AD member to the Source Instance,the merged DC.

This protocol does not use the delegation process as DCs are supposed to have the capacity to discover and communicate with each other directly.

DC Merge Ready and DC Merge Commit messages are also AD protected in order to confirm that both the Merging and the Merged DC actually know the AD Secret.

In addition to elements given in clause 6.3.6.4.1, the DC Merge Commit message current internal record of the merging DC. This allows the Merged DC to add the DC_local_id of the merging DC to the list of merged DC_local_ids in the internal record.

The DC Merged message, exchanged between the security controls of the DCs involved in the protocol, carries the ADSE values and the list of CICF identifiers that are transferred to the Merged DC so that they are integrated in its new ADSE counts and list.

In addition to codes given in clause 6.3.6.4.1, error codes associated with DC Merge Begin are:

- 'Sending Instance is not a member of any AD' if the Source Instance is a Blank Instance.
- 'Sending Instance is not Domain Controller' can occur if the Source Instance is not a DC and then cannot Merge.

The only additional error code associated with DC Merge Ready is 'Sending Instance is not a Domain Controller' if the replying CPCM Instance is not a DC.

The Error codes associated with DC Merge Commit are:

- 'Sending Instance is not a Domain Controller' if the Source CPCM Instance is not a DC. This may happen if the Begin message was not actually sent by that CPCM Instance.
- 'Internal_record with bad signature' if the received domain internal data record is corrupted.
- 'Inconsistent internal_record' if the received AD internal record is not consistent: For instance, this will be the case if it was not generated by the source CPCM Instance.
- 'Outdated internal_record' if the received domain internal data record is known to be out of date, usually because the Merged DC already has a copy of a later record.

Error codes associated with DC Merged are:

- 'Unexpected ADSE values' if the current received values are not internally consistent (e.g. if total_count is not the sum of remote_count and local_count) or if any other problem is detected.
- 'Unexpected list of CICFs that Joined Remotely' if the received list seems inconsistent (number of elements is different from remote_count, identifier of the peer Instance is in the list).

6.3.6.4.7 DC Rebalance

This protocol is used to when the user decides that there is a need to transfer some ADSE counts from one DC to another. It is also used to transfer identifiers of CICFs that Remotely Joined, thus allowing the possibility of gaining remote counts (if an Instance becomes Local to the DC that has this identifier in its list, it will be removed from the list and will be re-counted as a local join, thus allowing more Remote joins). The counts and the CICF identifiers are transferred from the Sink Instance to the Source Instance. If some counts or some CICF identifiers need to be transferred in the reverse sense, another protocol needs to be run.

This protocol does not use the delegation process as DCs are supposed to have the capacity to discover and communicate with each other directly.

DC Rebalance Ready and DC Rebalance Commit messages are also AD protected in order to confirm that both the rebalancing and the rebalanced DC actually know the AD Secret.

In addition to elements given in clause 6.3.6.4.1, the DC Rebalance Ready message carries the current ADSE values and the list of CICFs that Joined Remotely in order for the rebalancing DC to be able to select the ADSE counts, ceilings and the CICF identifiers that are to be rebalanced.

The DC Rebalance Commit message carries the ADSE values and the list of CICFs identifiers that are to be transferred. Values are positive and need to be lower than or equal to the values received in the DC Rebalance Ready message.

The Rebalance DC message, exchanged between the security controls of the DC involved in the protocol, carries the ADSE values and the list of CICF identifiers that are actually transferred. These values are generally the same as the ones requested in the DC Rebalance Commit message but might be lower (or carrying fewer CICF identifiers) if the security control of the Sink DC decided so.

The Rebalance DC message, exchanged between the security controls of the DC involved in the protocol, carries the ADSE values and the list of CICFs identifiers that are actually transferred. These values are generally the same as the ones requested in the DC Rebalance Commit message but might be lower (or carrying fewer CICF identifiers) if the security control of the Sink DC decided so.

In addition to codes given in clause 6.3.6.4.1, error codes associated with DC Rebalance Begin are:

- 'Sending Instance is not a member of any AD' if the Source Instance is a Blank Instance.
- 'Sending Instance is not Domain Controller' if the Source Instance is not a DC.

Error codes associated with DC Rebalance Ready are:

- 'Sending Instance is not a Domain Controller' if the replying CPCM Instance is not a DC.
- 'Unexpected ADSE values' if the current received values are not internally consistent (e.g. if total_count is not the sum of remote_count and local_count) or if any other problem is detected.
- 'Unexpected list of CICFs that Joined Remotely' if the received list seems inconsistent (number of elements is different from remote_count, identifier of the peer Instance is in the list).

Error codes associated with DC Rebalance Commit are:

- 'Sending Instance is not a Domain Controller' if the Source CPCM Instance is not a DC. This may happen if the Begin message was not actually sent by that CPCM Instance.
- 'Bad ADSE counts requested' if at least one requested ADSE count exceed the value of the Sink DC Controller or if the requested ADSE counts are not consistent or have higher values than the one sent in the previous message. In this case, The Sink DC controller has the possibility not to send any error but to propose consistent ADSE counts in the DC Rebalance message.
- 'Bad CICF identifier requested' if at least one of the requested CICF identifiers is not in the list of the Sink DC. In this case, The Sink DC controller has the possibility not to send any error but to directly remove the corresponding CICFs identifiers from the requested list.

Error codes associated with Rebalance DC are:

- 'Unexpected ADSE values' if the transferred ADSE counts do not correspond to the requested one. In this case, the source DC may also not generate any error and may accept the received counts. If the error is generated, the DC Rebalance protocol is stopped.
- 'Unexpected list of CICFs that Joined Remotely' if the transferred CICF identifiers do not correspond to the requested ones. In this case, the source DC may also not generate any error and may accept the received identifiers. If the error is generated, the DC Rebalance protocol is stopped.

6.3.6.5 ADMAAA

An ADMAAA, if any, is set up by the C&R regime. The way this contact is done is specified by the C&R regime.

This protocol can be used whenever the ADM Agent (ADMAAA) authorization is needed to perform an ADM protocol. In most cases, this will happen because one ADSE ceiling has been reached or, in case of Joining, the quorum test did not pass. The request message is sent by a Domain Controller.

NOTE: This protocol is only necessary when the ADMAAA functionality resides in a different device, i.e. outside the DC. This does not preclude the inclusion of ADMAAA functionality within a DC device, such as a CA or DRM component. Communications between CPCM and an internal ADM AAA are an implementation choice subject to C&R regime, and are out of scope of this multi-part deliverable.

The request includes the device and the domain identifiers, the current ADSE values, the status of the requesting Instance which is expected to be a Domain Controller, the performed ADM protocol and the reason why it cannot be performed locally (AD condition).

The ADMAAA may run additional protocols to make its decision. These protocols are outside of the scope of this multi-part deliverable.

EXAMPLE 1: A C&R Regime provides a web-based mechanism for granting ADSE exemptions. The protocol between the local ADMAAA function and the web based server is proprietary.

All protocol messages are authenticated using the SAC.

If the ADMAAA refuses the request, error 'Rejection by ADMAAA' is generated. Other specific error messages to the request are:

- 'Sending Instance is not Domain Controller' if the requesting Instance is not a DC
- 'The indicated AD has been revoked' if the AD has been revoked
- 'Certificate-message mismatch' if the status information in the request message does not match the certificate

If the request is accepted, the ADMAAA sends a message ADMAAA tool response carrying new ADSE values for the DC. It may carry mostly the same values as the original ones, but typically the ceiling that has been reached will have been increased (typically by one). It may also consist of exactly the same values if the protocol failed because of a quorum test failure, but the ADM AAA is willing to treat the quorum test as having been passed. Finally, wider grants may be given (e.g. more than one ceiling increased by two or more). The requesting CPCM Instance will update its ADSE values accordingly and proceed with the ADM protocol.

EXAMPLE 2: A CPCM Instance is attempting to Remotely Join an AD. Remote ceiling has been reached. The quorum test is not run for this Remote Instance, so the DC makes a request to the ADM AAA. The C&R regime has approved an alternative proprietary mechanism to validate such a Join using an account on a web server. This mechanism approves the Join, and the system proceeds as if the remote ceiling had not been reached. The ADSE counts remain unchanged.

The only specific error code to this message is 'Unexpected ADSE values' that may be generated if the returned ADSE values are lower than the original ones. In this case, two behaviours are possible (unless one is specifically mandated by the applicable C&R regime). Either the instance updates its ADSE values and proceeds with the protocol, or it generates this error code but does not proceed with the protocol and does not update ADSE values.

6.3.7 CPCM Extension Protocol

CPCM Extension messages are used to communicate information to or between CPCM Instances that implement a given CPCM Extension. See TS 102 825-14 [i.6] for details of CPCM Extensions.

The CPCM Extension message is intended for use by extensions that are added by the DVB or ETSI. Proprietary (non-standardised) protocols, whether for proprietary extensions or otherwise, should use Private Elements in CPCM Messages or Private Protocols instead.

6.3.8 Private Extension Protocol

Private messages can be used by implementers who wish to communicate proprietary information between CPCM Instances beyond that which is covered by the CPCM specifications.

It is also possible to attach private information to standard CPCM messages. Private messages should only be used when there is no suitable CPCM Message available for carriage of this information.

It is good practice to determine early whether the other Instance involved in the communication is able to handle these private messages. This can be done using a simple handshake of private messages. If the response is an error with "Unknown private identifier", the requesting instance immediately knows that the private protocol is not possible.

Private protocols may rely on the CPCM SAC or AD secret to encrypt their message contents, or they may provide their own means of protection. In the latter case it may be easier to send the private message not using any CPCM protection (i.e. CPCM message will not be signalled as encrypted or authenticated) and rely on the proprietary protection mechanism. This reduces the number of potential CPCM error conditions (e.g. SAC not established).

Private messages should have no effect on CPCM State machines. It is also unsafe to assume that they will only arrive in specific states. Handlers for the arrival of private messages should be provided in all states of the instance, even if they only provide error responses. If the private protocol requires multiple states, the implementation should define private error codes capable of reporting incorrect state/message combinations, as is done for the CPCM protocols themselves.

6.4 Content Management

The CPCM System effects Content management through the use of a common set of Usage Rules for all content within the CPCM System. Content which is external to the CPCM System is referred to as Input Content.

Input Content is Acquired by the CPCM System, along with its associated input usage rights, which are mapped onto the equivalent CPCM Authorised Usage rights, before it can become CPCM Content.

Every route for content Acquisition into CPCM will have a mapping of that route's set of Usage Rules into the common CPCM Usage Rules.

Once inside the CPCM System, CPCM Content can be Stored or Processed as permitted by the Authorised Usage rules bound to the Content.

CPCM Content leaves the CPCM System by being Consumed or Exported, at which point the CPCM Content's Authorised Usage states may be changed, according to the Usage Rules associated with the Content.

Although the basic framework for CPCM Content management is largely described in terms of the five CPCM abstract Functional Entities; Acquisition, Storage, Processing, Consumption and Export, other aspects of CPCM have a bearing on how Content is actually managed; these aspects include how CPCM operates within a home network ecosystem, how Content is moved across application-specific physical interfaces, the influence of the CPCM Security Toolbox on Content transactions, the handling of Content Licenses and the relationship between CPCM Device capabilities as defined within the CPCM Device Certificate and the associated Authorised Usage of the CPCM Content for each CPCM Device involved in Content exchange.

The following clauses describe in more detail the processes and issues involved in Content Discovery Exchange, and the specific behaviours of the five CPCM Functional Entities.

6.4.1 Content Discovery and Exchange

CPCM will be part of home network ecosystems capable of hosting both CPCM Devices and Content and non-CPCM devices and their content. CPCM Devices will use discovery mechanisms provided by the home network ecosystem in order to make the CPCM Content available to the user. In order to do this, there are CPCM-specific device attributes within the home network discovery process, and these attributes are stored in a secure manner within the CPCM Instance Certificate (CIC). The attributes will be replicated in the home network device discovery process, most likely as an extension to the normal home network device attributes, or possibly as a separate CPCM-specific discovery process.

6.4.1.1 Visibility of Content

Methods to ensure visibility of CPCM Content within the home network are not defined within the CPCM specification, but should be achieved through the standard home network navigation and directory interfaces, where CPCM content can be listed alongside other network content, or in separate CPCM Content directories.

When a CPCM Device has control over CPCM Content items it should advertise availability of that Content to the user via the CPCM Device's user interface, if present. Where CPCM Devices reside on a network, each CPCM Device should be able to discover the presence of CPCM Content maintained by other CPCM Devices by enquiring their respective CPCM Content directory services.

In the case of CPCM Content residing on a non-CPCM device such as a bit-bucket, the device hosting the Content is responsible for advertising availability of the Content using standard Content Discovery protocols, for example as defined for the DVB-HN.

In the case of self-contained CPCM Content (i.e., CPCM Content and its associated in-bound Content License, protected by the AD Secret) any number of CPCM Instances can be aware of such content and this will enable it to be discovered by other CPCM Devices in the home network and/or CPCM System.

Non-CPCM devices may also discover CPCM content in order to perform basic operations on it such as delete, copy or a Move. If the CPCM Content's Authorized Usage allows this, then the home network ecosystem's content management functions will request these to be executed by CPCM.

It is important that there should be user-friendly visibility (iconic or otherwise) of relevant CPCM Content properties through the network content guide / directory facilities, particularly with respect to Authorised Usage attributes.

Valuable data would include key Usage State Information (USI), relevant Auxiliary Data and the AD Identifier, showing which AD the content item is bound to. These properties can also similarly be used to filter the listing of Content items that are not readily accessible for whatever reason.

An implementation option for Acquisition Points and Export Points is to advertise respectively the availability of non-CPCM content that the CPCM functional entity may Acquire to CPCM or of CPCM Content that the CPCM functional entity could Export, subject to the usage rules applicable to the content.

6.4.1.2 Inter-Device Content exchange

Any implementation of a CPCM Device as a home network application will have to manage content transport requests, and the associated CPCM Instance will manage the corresponding CPCM Content Licence handling. It is the responsibility of the CPCM Device to ensure that there is a correct and consistent mapping between the home network ecosystem and the CPCM System.

It is important to remember that CPCM Content operations are communicated independently of any equivalent home network system signalling, in the CPCM Content Licence Exchange protocols. In this sense, CPCM is essentially decoupled from the home network ecosystem. This improves the security of CPCM Content operations, and at the same time ensures that there is a single common scheme for CPCM Content Management which all home network ecosystems supporting CPCM will map to.

6.4.2 Functional Entity Behaviour

The following clause describes each CPCM Functional Entity in terms of the CPCM Content management issues relevant to them.

6.4.2.1 Acquisition

When Input Content is Acquired by an Acquisition Point, A CPCM Content Licence is generated. If the CPCM Content is to be delivered to another CPCM Instance, USI instantiation in the Content Licence may be different from the originally received USI.

If the CPCM Content is not marked DNCS, it is then scrambled by applying the chosen descrambling_key and CPCM_descrambler_information as stored in the Content Licence. Associated CPCM Auxiliary Data, if present, will include the original usage rules signalling and other data as applicable.

A CPCM Instance that Acquires a Content item will then advertise the availability of the newly created CPCM Content item if the Content item is intended to be accessible by other CPCM Instances.

6.4.2.2 Processing

When receiving CPCM Content, a Processing Entity will verify the Content Licence, enforce the USI, and then check whether it is authorised to send the Processed CPCM Content to the Destination CPCM Instance.

Once these steps are completed, it will then process the CPCM Content. This processing may require CPCM Content descrambling. In this case, once processing has been completed, the Content will be re-scrambled. It may be that there is a requirement at this point to change the descrambling key, and a new key will be set to a random value and a new Content Licence is then issued.

- Amongst other fields, a Processing Entity may change the `descrambler_information` and `descrambling_key` fields in the Content Licence, and some fields in Auxiliary Data such as CLID or private data. In such cases it will need to update the CPCM Auxiliary Data digest. The Processing Entity may also change the C&R regime mask and the USI if permitted by its C&R regime, for example when updating the USI just before passing the content to a CPCM Instance which implements a different C&R regime.

- On completion of its processing tasks, the Processing Entity may then also send the processed CPCM Content to the Destination CPCM Instance.

6.4.2.3 Export

When receiving CPCM Content, an Export Point will verify the Content Licence, and enforce the USI conditions.

EXAMPLE: If image_constraint is asserted and the content is high resolution, the Export Point constrains the resolution to standard definition, via a suitable Processing Entity.

If Exporting is authorised to a Trusted CPS, a Controlled CPS or beyond CPCM Trust is authorized and if Content is not marked DNCS, the Content will be descrambled.

Once these steps have been completed, the Content will be Exported. In doing so, the exporting CPCM Instance will not advertise the availability of that content item to other CPCM Instances but may advertise it as content protected by the Export CPS.

The exporting CPCM Instance might also, depending upon the usage rules or on the user interaction, discard the obtained Content Licence for that Content item once Export has been completed.

Exported Content can be transported over the home network or over a CPCM application-specific interface.

Where a CPCM Instance Exports a CPCM Content to another CPS, it is required to map certain fields of the Content Licence and/or CPCM Auxiliary Data to that other CPS as defined by the associated C&R regime.

6.4.2.4 Consumption

When receiving CPCM content, a Consumption Point will verify the Content Licence and enforce the USI conditions. If Consumption is authorised, then the Content will be descrambled if not marked DNCS.

The Content can then be Consumed. The CPCM Instance that Consumes a CPCM Content item will not advertise the availability of that Content item to other CPCM Instances. It will also discard the obtained Content Licence for that Content item once Consumption has been completed. A renewed Consumption or an Export operation on the same Content item will happen only after the associated Content Licence has been re-obtained from the Source CPCM Instance.

A CPCM Instance that Outputs a CPCM Content item to a Consumption Output will map certain fields of the Content Licence and/or CPCM Auxiliary Data to that other CPS as defined by the associated C&R regime.

EXAMPLE: If content is Consumed through an analogue output, content resolution may be downsized if USI image_constrain is asserted.

6.4.2.5 Storage

When receiving CPCM content, a Storage Entity will verify the Content Licence and enforce the USI, and if required will change certain allowed elements of the associated USI to indicate that a new Copy has been generated. The Storage Entity is also responsible for protecting the Content Licence it needed, for example if the CL was received through a SAC.

Once this is completed, the Storage Entity will Store the CPCM Content item.

When delivering CPCM retrieved Content to another Entity, a Storage Entity will verify the Content Licence, then enforce the associated USI and will issue a new Content Licence if needed. If the Content Licence was protected using Device key, the Storage Entity will then, protect the Content Licence.

Once these steps have been completed, the Storage Entity can deliver the CPCM retrieved Content.

If the Content Item is intended to be accessible by other CPCM Instances, then the CPCM Instance that Stores the CPCM Content item will advertise the Content's availability to other CPCM Instances.

6.4.3 USI enforcement

6.4.3.1 Introduction

In TS 102 825-4 [i.9], USI enforcement is dealt with in the context of the normative behaviour of sink and source devices. This clause explains how the actions performed by these devices actually achieve USI enforcement.

Even if enforcement is described separately for each USI, all the USI included in a Content Licence have to be obeyed in order to be authorized to manage associated Content.

Restrictions are described in term of functional entity. CPCM Instances having several roles are able to handle or exchange of a Content as soon as one of its functional entities is authorized but only actions authorized by the USI will be permitted.

EXAMPLE: A CPCM Instance that is a Storage Entity and a Consumption Point may receive content marked 'not Viewable' for storage even if Consumption Points are banned to receive such content. It will however not be able to Consume it.

Keys and rights obtained from a Content Licence or from protocols run to enforce the licence are only valid as long as the CPCM Instance is still handling the content. They can generally not be re-used for a future session as the Content Licence enforcement has to be performed each time. Implementations have to prevent the re-use of such keys and rights. A good practice is to erase them as soon is the Instance stops dealing with the Content Item.

6.4.3.2 Copy Control Enforcement

6.4.3.2.1 Copy Control Not Asserted Enforcement

No specific action has to be performed in order to enforce CCNA content. If the CL is AD protected, content and its associated CL may be sent to anyone, including non CPCM devices. If the CL is device secret protected, content may also be sent for storage to any device, but content consumption will necessarily be done through the device that initially stored the content, if the CL was not passed through a SAC by the storage device.

6.4.3.2.2 Copy Once, Copy No more Enforcement

Copy Once content may only be sourced from two types of CPCM Instance: an Acquisition Point or a Processing Entity which has received it from another AP or PE.

An AP may only source one Instance of Copy Once content. All other instances of the CL are to be marked copy no more. The CPCM Instance that will eventually receive such a CL will necessarily be a CP or an EP and thus associated USI will be respected.

A storage entity receiving copy once content re-marks it as copy no more before actually recording it. It may then source the content to any CPCM Instance with the exception of Storage Entities. For Storage Entities, the Move protocol has to be used (see clause 6.4.6).

6.4.3.2.3 Copy Never Enforcement

It is expected that all CPCM content will allow trick modes, including copy never content. For the latter, only one storage entity will be enabled to store the content, to avoid easy circumvention of the intended usage. Hence, the first Storage Entity that stores the Content has to re-mark it as 'Zero Retention Asserted'. Trick plays will have to be run from that Storage Entity.

Recorded copy never content can only be kept up to the limit fixed by the applicable C&R regime. It may be moved using the Move protocol.

Content already marked as 'Zero Retention Asserted' can never be re-marked as 'Zero Retention Not Asserted'.

6.4.3.3 Consumption Control Enforcement

6.4.3.3.1 Viewable

If not asserted, this USI only forbids to Consume or Export the content. All other actions (including Processing, Storing and Copying) are permitted, unless otherwise restricted by other USI.

6.4.3.3.2 View Window and View Period Enforcement

VPA content is considered as 'Copy No More' content before the View Period starts. This is the reason why it has to be Moved between Storage Entities and cannot be simply copied. If several instances of the content licence were existing, it would not be possible to start the View Period for all the content licences (e.g. because one is disconnected at that time) and could result in the USI circumvention.

Copying VPA content is however permitted but is considered to be Consumption and hence causes the start of the View Period. Implementers are strongly advised to warn the user that this will start the View Period.

Content may be marked simultaneously 'View Window Activated' and 'View Period Activated'. In this case, a n attempt to Consume or Copy will start the View Period even if the View Window has not yet started. The following behaviour is advised:

- If the user tries to consume or export the content, implementations should refuse to perform the action, based on the fact the View Window has not yet started
- If the user tries to copy the content, implementations should warn the user that this will start the View Period while the View Window has not yet started so that the user will not be able to consume the content immediately. If the View Period ends before the View Window starts, it should refuse to perform the action as the content will never become playable. If the View Period ends after the View Window, implementations should clearly inform the user about the new View Window that would apply if the user confirms his request.

Due to the complexity of the above, some implementers may reasonably decide to disable copies outside of the View Window.

EXAMPLE: On October 31st, Bob purchases a content item with a View Period of 48 hours and a View Window set to November. Bob will travel the next day and wants to transfer the movie to its PMP so that he can watch it in the train. He informs his wife Alice that she will be able to see the content item for the next two days.

6.4.3.3.3 Simultaneous View Count Enforcement

This USI applies only to direct/live consumption of content, i.e. content that is currently being Acquired into CPCM. It restricts the number of Consumption Points and Export Points that may have access to that live/direct content. It does not affect the ability to Store or Process the Content.

The Acquisition Point is responsible for enforcing this limit. It maintains the count and increments it each time it delivers a Content Licence to a Consumption Point or to an Export Point. The count is not affected for Storage Entities. For Processing Entities and CPCM Instances which implement several functionalities, the Acquisition Point counts the CL delivery if it is allowed for Consumption or Export, i.e. if the CPCM_content_handling_operation field allows for Consumption or Export in the message delivering the CL; CPCM get CL response or CPCM put CL calls.

EXAMPLE 1: If the Acquisition Point pushes, using CPCM put CL call, the Content Licence for storage to a CPCM Instance that is both a Storage Entity and a Consumption Point, it will allow only Storage in the CPCM_content_handling_operation field and thus will not count this delivery. If a Consumption Point requests a CL through a Processing Entity, the CPCM_content_handling_operation field will indicate that the content is requested for Consumption and thus will count the CL delivery.

When the limit is reached, the Acquisition Point may not deliver a new CL until it has checked that at least one CL that was previously delivered is no longer in use (see explanations below on the 'follow me' capability).

SVCA enforcement provides support for the two following cases:

- Delayed viewing: content may be first sent to a storage entity, or a bit bucket, and then played with some delay. If the content is direct/live, the SVCA restriction applies to this delayed viewing. This is achieved by mandating the running of the CPCM Content Operation Permission protocol (see clause 6.3.5.3) when receiving the CL from a CPCM Instance that did not initially create the CL (which is known through the CLC field). Permission will be granted if the SVCA ceiling has not yet been reached.

- Follow me scenario: once the ceiling has been reached, the specification enables the consumer to change the set of Consumption or Export Points that are allowed to view the content. This is achieved by running the Content Item Status protocol (see clause 6.3.5.4) that allows the Acquisition Point to verify which devices are currently Consuming or Exporting the Content Item. If a CP or an EP has stopped consuming it, a new permission may be granted to a CP or an EP requesting it.

The Content Operation Permission Protocol is run as soon as content is not received from the AP that created the CL. If the origin of the CL cannot be determined (this is the case when AD secret is used to protect the CL), the protocol is also run. Hence, it is recommended for an AP delivering the CL to a CP or an EP to use SAC protection for the CL. In other cases, AD protection or SAC protection may be used equally. The Content Item may also be Stored on a bit bucket, in which case AD protection is the only possibility.

The Content Item Status protocol is run each time a CL or permission is requested while the SVC ceiling has already been reached. The request is sent to all CPs and/or EPs that are currently allowed to view the content. Hence, the AP has to maintain such a list for each Content Item marked SVCA as long as the content is live or direct. If all the CPs and/or EPs respond that they are still Consuming or Exporting the content, this means that no new permission may be granted. If all the CPs and/or EPs respond and at least one response is negative, the AP updates its list of permitted CPs and EPs and may grant a new CL or a new permission.

EXAMPLE 2: SVC is set to 1 and the CL has been delivered for Consumption and Storage to a CPCM Instance that is a Storage Entity and a Consumption Point. The AP receives a new request for Consumption. It runs the Content Item Status Protocol with the CPCM Instance which then replies that it is now only Storing the Content. The AP may thus grant the permission to the other CPCM Instance.

Different behaviour is adopted if no response is received at all from at least one CP or EP. There may be two reasons for this:

- No response is received because the CP or EP is no longer connected to the network. This means that the CP or EP is no longer Consuming or Exporting the content and hence a new permission may be granted.

- No response is received because the request and/or the response have been deliberately blocked by an attacker in order to get extra permissions. There is no means for the AP to distinguish this case from the above one.

The specification provides means for the AP to grant permission in the first case while making the attack ineffective in the second case. To grant a new permission, the AP changes the content scrambling key and delivers the corresponding new CL to all CPs and EPs that responded positively, and to SEs to which content was delivered. If an attacker has blocked the message exchanges with a particular CP or EP, the latter will not have received the new CL and hence will stop Consuming or Exporting the content.

This behaviour allows SVCA to be enforced but the process is convoluted. Hence, it is recommended for an AP to run the Content Item Status protocol a second time with non-responding CPs and EPs to ensure that the lack of response is not due to other problems such as bad network connectivity.

Implementations may also choose to presume that non-responding Instances are still Consuming or Exporting the content and to behave accordingly. This will simplify the implementation and provides similar user experience when at least one CPCM Instance has replied it was no more consumed the content. However, user experience might be decreased in the other case.

Content Item Status protocol may also be run on a regular basis by the AP once the ceiling has been reached to be able to respond more rapidly to new CL or permission requests. In this case, the AP has to record, in its list, which Instances were not responding as it will have to change the scrambling key in order to deliver the corresponding permission. Another possible, preferred behaviour in this case is to consider that non-responding Instances are still consuming the content. Hence, if a new CL or permission is requested, the AP will grant a new CL only if Instances had previously answered negatively to the Content Item Status Protocol. Else, it will re-run the Content Item Status protocol and behave as described above.

A CP or an EP which stops consuming or exporting the content may also warn the AP of this change. This is recommended for CPCM Instances that will be disconnected from the network (e.g. because they are powered down) in order to avoid changes of content scrambling key.

When content is no more live or direct, the restriction no more applies. The Acquisition Point can deliver the content licence to any number of Consumption or Export Points. It also delivers a new Content Licence where SVCA is no more asserted to all Storage Entities to which a CL was previously delivered and to new Storage Entities requesting the CL.

6.4.3.4 Propagation Control Enforcement

Content propagation is permitted within CPCM if at least one of the following conditions is met:

- Viewing is permitted in the sink CPCM Instance because Viewing Propagation Information (i.e. VLAD, VGAD, VAD or VCPCM) is verified.
- Viewing is permitted in the sink CPCM Instance because Content is marked VLocal and Sink Instance is Local to the Source Instance.
- Moving is permitted to the sink CPCM Instance because Moving and Copying Propagation Information (i.e. MLAD, MGAD, MAD or MCPCM) is verified.
- Moving is permitted to the sink CPCM Instance because Content is marked MLocal and Sink Instance is Local to the Source Instance.

When the CL is pulled from another CPCM Instance, the intended actions, viewing, exporting, moving and/or copying, on content is given in the request message (see clause 6.3.5.2). Only the requested actions need to be verified.

EXAMPLE 1: If the CL is requested for Consumption, only Viewing Propagation Information and VLocal (if asserted) need to be verified.

If the content is pushed, or is requested for Exporting or Processing, both types of action have to be verified and the CL push message will indicate which of these actions are permitted. This has to be re-assessed and updated by intermediary Processing Entities, if any. If only one action is permitted, applicable compliance rules will govern the permission to Export.

When MLocal is asserted, there is no need to assess both Movement and copying Propagation Information and MLocal and similarly when VLocal is asserted, there is no need to assess both Viewing Propagation Information and VLocal since in both cases propagation is permitted as long as one is verified. The verification can be made in any order. For content whose CL is AD protected, it is recommended to first verify VPI or MCPI, as sending under VLocal or MLocal protection will require SAC protection and hence will require a re-build of a new CL (while VPI or MCPI will generally allow propagation under AD protection). The only exception is when content is marked MCPCM or VCPCM and is propagated to a different AD, but in this case there is no need to assess MLocal or VLocal (as MCPCM or VCPCM is more permissive).

Enforcement of propagation is shared between the Source and the Sink, depending upon the USI, the type of protection of the CL and whether the CL is pushed or pulled.

EXAMPLE 2: If proximity-restricted content is pulled through a Processing Entity, proximity check between the Source and the Sink will be performed by the Source before sending the CL while if the CL is pushed, the Sink will perform the proximity check.

6.4.3.4.1 MLAD/VLAD Enforcement

There are different scenarios under which MLAD/VLAD USI can be assessed:

- Content Licence is sent solely under the SAC protection. In this case, the source CPCM Instance performs a proximity test with the sink Instance and an AD membership protocol. If both succeed, the content can be sent. The sink device only checks that the content is not bound to a different AD than the one to which it belongs.
- Content Licence is sent under both the SAC and AD protection. In this case, the source Instance performs a proximity test with the sink CPCM Instance. If the test succeeds, the content can be sent. The sink device does not need to perform any additional check.
- In the two above cases, if the content is sent through a Processing Entity, the request indicates the final sink CPCM Instance to which the content will eventually be sent if the CL is pulled. The source CPCM Instance also performs a proximity test with this sink CPCM Instance. If the CL is pushed, the source CPCM Instance inserts its CIC identifier in the push_CL_message and the sink CPCM Instance also performs a proximity check with the source CPCM Instance. Each intermediate Processing Entity performs the tests described above, depending upon the content protection. The rationale for performing such additional verifications is to avoid the CL being sent Remotely through multiple Processing Entities that are Local to one another, but collectively form a chain that extends beyond the Local Environment.

- Content Licence is sent solely under the AD protection. In this case, the source CPCM instance inserts its CIC identifier in the last_CL_issuer field of the CL. Content can be sent to any type of CPCM Instance or to a non-CPCM device. When the content is received by the sink CPCM Instance, it performs a proximity test with the source CPCM Instance (which is identified through the last_CL_issuer field).

Proximity tests may be performed in advance, in which case they need not to be re-performed. The AD membership test cannot be skipped if the CL is sent under the SAC protection only.

MLAD/VLAD restriction always expires, possibly after a very long time. Source and CPCM Instances may then check whether the rule has expired before performing the proximity check, if it was not performed in advance, or if the latter failed.

6.4.3.4.2 MGAD/VGAD Enforcement

Geographically restricted content is always available Locally, even when not in the designated geographical footprint. Thus, it is recommended to firstly assess whether the content source is Local or not.

In contrast to the MLocal/Vlocal case, the proximity test is never performed by the Source CPCM Instance. Indeed, even if such a test fails, the transfer would be permitted as the Sink CPCM Instance might be within the authorized geographical footprint. Hence, proximity is always checked by the Sink CPCM Instance for MGAD/VGAD.

The Source CPCM Instance only enforces AD restriction for MGAD/VGAD. To this end, it uses either AD protection for the CL, possibly together with SAC protection, or it uses SAC protection but it first performs an AD membership challenge protocol. If the CL is solely under AD protection, it inserts its CIC identifier in the CL. Otherwise it inserts its identifier in the message carrying the CL. It does not need to check whether the restriction still applies or not.

The Sink CPCM Instance then behaves as follows:

- If it knows the CL is received from a Local Source (which is identified using either last_CL_issuer or the source identifier in the message), it can handle the CL.
- Else it may check whether the geographical restriction still applies or try to perform the proximity check.
- If the test failed, it enforces the geographical restriction. If it is not geographically aware, it needs to find another Local CPCM Instance that is geographically aware using the CPCM Discovery Protocol (see clause 6.3.5.10). If no such CPCM Instance is discovered, it cannot determine whether it is within the authorized geographical footprint and since it received the content Remotely, thus it cannot handle the Content.
- If the CL was received solely under SAC protection, it also verifies that the CL is bound to the AD to which it belongs and performs an AD membership protocol to check the originating Instance belongs to the AD.

6.4.3.4.3 MAD/VAD Enforcement

The Source CPCM Instance only enforces AD restriction. To this end, it either uses AD protection for the CL (possibly together with SAC protection) or it uses SAC protection but firstly performs an AD membership challenge protocol. If the CL is solely under AD protection, then it inserts its CIC identifier in the CL. Else, it inserts its identifier in the message carrying the CL.

The Sink CPCM Instance does not need to perform any verification, unless the CL was received solely under SAC protection, in which case it verifies the CL is bound to the AD to which it belongs and performs an AD membership protocol to check the originating Instance belongs to the AD.

6.4.3.4.4 MCPCM/VCPCM Enforcement

There is no verification to enforce the USI in this case. However, if the CL is protected by the AD secret (with or without SAC protection), the Source CPCM Instance needs first to check that the Sink Instance belongs to the same AD else it would not be able to access to the content. If it does not belong to the same AD, it will need to switch the CL to SAC only protection.

6.4.3.4.5 MLocal/VLocal Enforcement

The Source CPCM Instance has to perform a proximity check with the Sink Instance (if this proximity check has not been performed in advance).

In addition, if the content is sent through one or more Processing Entities and if the CL is pulled, the Source CPCM Instance has to perform a proximity check with the final destination of the CL. If the CL is pushed then the Sink CPCM Instance will perform the check with the source CPCM Instance indicated in the push_CL_message.

6.4.3.5 Export and Consumption Control Enforcement

Export and Consumption Control is only meaningful to Export and Consumption Points. All other CPCM entities may ignore it. Thus only sink entities need to look at these USI.

6.4.3.6 Ancillary Control Enforcement

DNCS marked content is never encrypted by CPCM. Content that is not marked DNCS is always scrambled using LSA. Thus, the AP acquiring non-DNCS content needs to be LSA capable or it cannot Acquire the content.

6.4.3.7 Remote Access Rules

In CPCM, geographical and proximity restrictions always expire once at least one of the Remote Access Rules applies.

6.4.3.7.1 Post Record

For content whose remote access rule is 'Remote Access Post Record', content cannot be moved and/or viewed outside of the restricted area as long as the content is still direct or live.

Thus, when content is marked with this USI, and if no other RAR rules apply, the Remote Sink CPCM Instance needs first to seek permission from the Acquisition Point. This is done using the Content Operation Permission Protocol (see clause 6.3.5.3). If the content is no longer direct or live, a new version of the CL is created with all remote access rules flags unasserted and the content is marked as MAD and/or as VAD, and is delivered using CPCM_put_CL_message. Otherwise access is denied.

If another RAR applies, content is accessible Remotely by the application of this rule, and there is no need to issue a new CL.

There are two cases when the Content Operation Permission Protocol will be run:

- A CPCM Instance wants to transfer content Remotely under SAC protection: it runs the protocol to assess whether the transfer is permitted or not.
- A CPCM instance receives the Content Item under AD Secret protection. It performs a Proximity test with the Acquisition Point that delivered the CL (identifiable from the CLC field). If it is not Remote, it may access the content. Otherwise it runs the protocol to check whether permission is granted or not.

6.4.3.7.2 Post date/time (moving window and immediate)

This RAR has to be assessed under two circumstances:

- A CPCM Instance wants to transfer content Remotely under SAC protection.
- A CPCM instance receives the Content Item under AD Secret protection and is Remote from the AP that created the CL.

In both cases, the CPCM Instance has to access Secure Absolute Time in order to know whether it can transfer or manage the content. If it is not itself absolute time aware, it runs the CPCM Secure Time Protocol (see clause 6.3.5.6), to obtain it. If no secure source of absolute time is available, or if the time criterion is not met, it cannot transfer or manage the content.

An Acquisition Point may also deliver a new CL where all RAR flags have been cleared and MAD and/or VAD are asserted to avoid blocking Remote Access to the content item in the absence of a secure absolute time source.

6.4.3.8 Copy No More and View Period Activated special consideration

As mentioned in TS 102 825-4 [i.9], a particular attention is to be given by implementers in the enforcement of VPA USI and Movement of CNM content. The specific threat to deal with is the ability for an attacker to copy back a previous Content Licence (possibly with the associated content) to the storage associated with the CPCM Instance:

- If the content is marked 'Copy No More' and the associated Content Licence was Moved, restoring a Content Licence associated with content before the Move occurred may result in a valid copy of the content, and hence a second copy of the content, violating thus the USI. The same stands for a content marked 'View Period Activated' that would have been Moved.

- If the content is marked 'View Period Activated' and the content has been consumed, copied or exported, content was re-marked as 'View Window Activated' with relevant time-window. Restoring the Content Licence before the consumption, export or copy of the content occurred may result in a new valid content marked 'View Period Asserted' where consumption, export or copy did not yet occur, hence violating the USI.

Implementations have to take into account this threat and prevent the above attack succeeding.

A possibility to handle this threat would be for the implementation to build a list of all the 'Copy No More' or 'View Period Activated' Content Licences that have been Moved to another CPCM Instances and of all 'View Period Activated' Content Licences for which the View Period has started. The storage of this list has obviously to be secured:

- This list could be stored in secure memory. If this solution is chosen, implementations should have sufficient secure memory for storing the complete history of such licences. This approach is not practical as it potentially requires an infinite amount of secure memory.

- The list may be stored authenticated in standard memory. The key used to authenticate the list has to be protected in integrity and renewed each time the list changes, in order to prevent restoring an older version of the list.

- The list may be stored authenticated in standard memory and include an index that is incremented by one at each list update. The current value of the index is kept in secure memory.

Another possibility is to keep a list of Content Licences of 'Copy No More' content and 'View Period Activated'' content that are still valid. The list may be protected using one of the above methods.

Yet another possibility is for implementations to have a specific key dedicated to the security of 'Copy No More' or 'View Period Activated' content and change this key each time one of associated Content Licence is not valid anymore. All the associated Content Licences have to be immediately re-protected with the new key.

6.4.4 Content Scrambling

6.4.4.1 Support of key changes by the Acquisition Point

As specified in TS 102 825-4 [i.9], the CPCM content management scheme provides a mechanism that allows forcing changes of Content Scrambling Key (CSK). Thus, a Crypto-Period (CrP) controlling duration of CSK use while encrypting content, can be applied by a Source CPCM Instance if necessary, by repeated application of this mechanism.

This clause discusses implementation aspects in relation with the management of CrP. In particular, it deals with the situation where the external system delivering content to the CPCM system through the CPCM AP uses CrP. Different CPCM compliant implementation choices are explained. The goal is to bring out their impact on:

- Cutting of Acquired content into CPCM Content Items

- CLID management

- CL repetition and provision scheme

- Management of Simultaneous View Count feature

- Management of Remote Access features

What follows is built on the paradigm where content is acquired as broadcasted DVB event, and is associated with a DVB event ID. The term 'CAS-CrP' will be used to designate the CrP ruling the encryption of the acquired event, and which is originated by the Conditional Access System. The term 'CPCM-CrP' will be used to designate the CrP ruling the encryption of the corresponding CPCM content inside the CPCM system; it may be abusive, since it designates the duration of a CSK appliance, not necessarily repeated as the term 'period' suggests. This paradigm easily extends to acquisition of content from any external system.

6.4.4.2 Case 1: One CSK per event

It is supposed that during Acquisition each received event is scrambled to enter the CPCM system with one unique CSK. This may be a C&R regime requirement or a free AP implementation choice.

In this case:

- Mapping of event to CPCM Content Item: The AP descrambles each event segment corresponding to one CAS-CrP and re-scrambles it with the same CSK during the whole event. The event is mapped on one unique CPCM Content Item, to the exception of the Simultaneous View Count management as explained below. At each event start, signalled by a DVB-event-ID change, the AP issues a new CL; this CL carries the USI mapped from the Usage Rules signalled by the CAS with the event (see clause 10.1.3) and the CSK used to scramble the related CPCM Content Item.

- CLID management: CPCM Content Item transitions happen at event transitions, to the exception of the SVC management as explained below. If the AP assigns the same CLID to the CL of the new CPCM Content Item as to the previous one, it becomes difficult to correctly respond to subsequent CL-related messaging (see clause 6.4.4.7). It is then recommended to change the CLID from one event to the other. The 'DVB Broadcast Event and Acquisition Device' CLID allocation scheme is well adapted to allocate a new CLID to the new Content Item (i.e. event or Content). The uniqueness of DVB-event-ID will guarantee a one-to-one correspondence between CLID and Content Item. The freely allocable byte may remain null, and, since DVB-event-ID uniqueness is managed by the delivery system, the AP needs no additional workload or resource for this task.

- Odd-Even transition: The *odd_even_key_indicator* bit in the *CPCM_descrambler_information* structure of the CL should be used to ensure smooth event transition. It is then recommended to change the key parity from one event to another, and to send the new CL in advance of the new event. In case of the MPEG-2 TS adaptation layer, the AP will set the parity bit in the *transport_scrambling_control* bits of the TS header to be equal to the *odd_even_key_indicator* of the CL carrying the CSK currently in use. The parity bits needs then to be changed at Content Item transition, which occurs at event end.

- Management of Simultaneous View Count (see clause 6.4.3.3.3): SVCA control impacts the cutting of the event into CPCM Content Items. When the maximum view count is reached and the event is sent to a new CP or EP because some CPCM Instance did not respond to a *CPCM_get_content_item_status* message, a CSK change needs to be performed. Hence, the received event is split into two CPCM Content Items. The AP re-issues a CL with the new CSK. If a new CLID is computed, the CLID allocation mechanism should use the freely allocatable bytes of the CLID allocation scheme to differentiate the new CL. In case of the 'DVB Broadcast Event and Acquisition Device', one byte is allocatable, and hence 255 similar changes may be supported. Hence, the applicability of this schema depends on the number of switches performed by the user, in the 'Follow me' scenario (switching off one CP or EP before switching on another CP or EP), and on the maximum event duration, depending in turn on event delineation by the delivery system. Both parameters are unknown. If too many screen switches occurred, the AP should be able to warn the user or to change the applied CLID allocation scheme. It is then recommended to keep the same CLID at CSK change due to SVCA control during a given event (see clause 6.4.4.7).

- Management of RAR: If a CSK change occurred due to SVCA control as detailed above, and the event is submitted to Remote Access Post Recording condition, then at the end of the event the AP re-issues CL with the updated USI. If the related CPCM Content Items have different CLID, the AP can re-issue one CL per Content Item related to the event, each one with the correct CLID, and the applied CSK. This requires the AP stores CL generated for the CPCM Content Items of the current event. Else, the AP sends the updated CL for the last item, with the last CSK, and it relies on SE implementation to manage the update of the linked CL, i.e. each CL related to the same event.

6.4.4.3 Case 2: Several CSK per event

It is supposed that during Acquisition the applied CSK changes according to a specific CPCM-CrP and the duration of this CPCM-CrP is such that the forced CSK change occurs only a few times during one event (e.g. the CPCM-CrP is about 30 minutes). This may be a compliance regime requirement or a free AP implementation choice.

Since the CAS-CrP is characteristically of the order of magnitude of a few seconds, this case typically corresponds to situations where the CPCM-CrP is much greater than the CAS-CrP. This assumption can be made when CPCM-LSA is more secure, and/or uses larger keys than the delivery system (DVB-CSA in the paradigm). It is even justifiable if key lengths are comparable (e.g. with DVB-CSA3), because the threats addressed by CAS-CrP on one side (easiness of worldwide illegal sharing) and CPCM-CrP on the other side are very dissimilar.

In this case:

- Mapping of event to CPCM Content Item: The AP descrambles each event segment corresponding to one CAS-CrP and re-scrambles it with the same CSK during a given CPCM-CrP. The event is mapped to several CPCM Content Items. At each event start, signalled by a DVB-event-ID change, the AP issues a new CL; this CL carries the USI mapped from the Usage Rules signalled by the CAS with the event (see clause 10.1.3) and the CSK used to scramble the first CPCM Content Item. Then, after CPCM-CrP duration, the AP issues a new CL; this CL carries the same USI as before with the new, randomly generated, CSK used to scramble the new CPCM Content Item.

- CLID management: CPCM Content Item transitions happen at event transitions and at CSK changes with CPCM-CrP. The 'DVB Broadcast Event and Acquisition Device' CLID allocation scheme is well adapted to an event transition, with the advantages described in case 1. However, if the AP allocates a new CLID at a transition due to CSK change, the same allocation scheme can be used if the CPCM-CrP is large enough to guarantee a maximum of 256 CPCM Content Items per event. This in turn depends on the CPCM-CrP and on the maximum event duration, which is unknown. It is recommended to keep the same CLID at CSK change during a given event (see also clause 6.4.4.7).

- Odd-Even transition: The same odd-even transition mechanism as described in case 1 above should also be applied to CPCM Content Item transition during a given event.

- Management of Simultaneous View Count (see clause 6.4.3.3.3): SVCA control has the same impact as described in case 1 above; if CLID was changed with CPCM-CrP in addition to changes due to SVCA control, the applicability of the 'DVB Broadcast Event and Acquisition Device' CLID allocation scheme would even be more jeopardized. There is one more responsibility for the AP in this case: During a given event, at a CSK change due to CPCM-CrP, the AP needs to remember not to send the new CL with the new CSK to those CPCM instances that did not reply to the previous `CPCM_get_content_item_status` message.

- Management of RAR: If an event is submitted to Remote Access Post recording condition, then at the end of the event the AP re-issues one CL per CPCM Content Item related to the event. The same process as in case 1 may be applied.

6.4.4.4 Case 3: One CSK for several events

It is supposed that during Acquisition the applied CSK changes according to a specific CPCM-CrP, and the duration of this CPCM-CrP is such that a given CSK is likely to span several events (e.g. the CPCM-CrP lasts several hours). This may be a compliance regime requirement or a free AP implementation choice.

In this case:

- Mapping of event to CPCM Content Item: The AP descrambles each event segment corresponding to one CAS-CrP and re-scrambles it with the current CSK. If no CSK change occurs during a given event, the event should be handled as in case 1: At each event start, the AP issues a new CL, which encapsulates the current CSK. Thus, there is at least one CPCM Content Item per event. If a CSK-change occurs during an event, the remaining of the event is mapped on a second CPCM Content Item, bound to a second CL encapsulating the new CSK.

Other aspects (CLID management, Odd-Even transition, Management of Simultaneous View Count, and Management of RAR) are handled as explained in case 1 (if no CSK change occurs during the event) or case 2 (if a CSK change occurs during the event).

6.4.4.5 Case 4: Many CSK per event

It is supposed that during Acquisition the applied CSK changes according to a specific CPCM-CrP, and the duration of this CPCM-CrP are such that the forced CSK change occurs a large number of times during one event (e.g. the CPCM-CrP is about 10 seconds). This may be a compliance regime requirement or a free AP implementation choice.

Contrarily to case 2, this case typically corresponds to situations where there is an obligation for the AP to change the CSK with the CAS-CrP itself. As explained in case 2, this is unlikely to be required just for security reasons.

Basically, Mapping of event to CPCM Content Item, CLID management, Odd-Even transition, and Management of RAR can be handled as in case 2. Nonetheless, the very small duration of the CPCM-CrP has the following additional impacts:

- CLID management: It is not possible to use of the 'DVB Broadcast Event and Acquisition Device' CLID allocation scheme. Not changing CLID during a given event is strongly recommended.
- Management of Simultaneous View Count (see clause 6.4.3.3.3): SVCA control is simplified. The AP should just wait for the next CPCM-CrP, and at that time, it should no more stream the new Content Item and the new CL to these CP or EP having not responded to the previous *CPCM_get_content_item_status* message. Hence, no specific CSK change is required by SVCA control.

6.4.4.6 Case 5: Parity bits changed with CAS-CrP

It is supposed the AP is not capable of changing the *transport_scrambling_control* bits of the TS header. In this case, the AP would be forced to issue a new CL at each new CAS-CrP, so to set the *odd_even_key_indicator* bit in the *CPCM_descrambler_information* structure of the CL in a way consistent with the received TS header content.

But in order to avoid the complications of case 4, as well as to be able to fit other possible compliance requirement on CPCM-CrP, the AP can avoid a CSK change by proceeding as follows:

- The AP descrambles each event segment corresponding to one CAS-CrP and re-scrambles it with the same CSK as in case 1 or 2, depending on the CPCM-CrP.
- During a given CPCM-CrP, at each CAS-CrP the AP issues a new CL, with same CSK and USI, but with the *odd_even_key_indicator* bit switched so to be equal to the current value of the parity bit in the TS header.

This method does not impact other CPCM Content Item related management. In particular, CLID allocation, and CSK changes with CPCM-CrP and/or SVC control, can be performed as described in cases 1 to 4.

6.4.4.7 CLID of new CL

At the time of a CPCM Content Item transition, the AP may either assign the CLID of the previous CL to the new CL or allocate a new CLID. In the above examples, the AP issues new CL when a new Content Item has started. This happens at the event transition (all cases), at a CSK change in the context of Simultaneous View Count feature management (cases 1, 2 and 3), or at a CSK change forced by the CPCM-CrP (cases 2, 3 and 4).

The handled cases show that the use of the 'DVB Broadcast Event and Acquisition Device' CLID allocation scheme leads to uncertainty regarding the possibility to support as many CLID changes per event as needed by CSK changes due to the combined effect of CPCM-CrP and SVCA control. Hence, it is preferable to keep the same CLID in cases of CSK changes during the same event.

Keeping the same CLID from one item to the next, has no further impact on the AP function. Indeed, the practical use of CLID is to refer a specific CL in CL exchange messages. For instance, if a CP, an EP, or a PE requires a licence with a specific CLID, a response with the licence for the currently acquired CPCM Content Item will always be appropriate. If a SE requires a new CL with a *CPCM_get_new_CL* message for a previously acquired item, the USI is the crucial information that needs to be updated. The AP may either:

- Respond with any of the up-to-date CL for the same event, relying on the SE implementation (then, the AP can just use the CL of the currently acquired item). According to TS 102 825-4 [i.9], the SE should have either stored the items with the received related CL and chained them in some proper way, or re-encrypted the items into one unique item with same CSK. In both cases, the SE should be able to replace the USI of (each) stored CL with the one sent by the AP with the CL response (or any *CPCM_put_CL message*), and ignores the CSK in the CL response.

- Respond with a CL built from current CL, in which parity indicator and CSK are replaced by the ones of the CL in the *CPCM_get_new_CL* message.

Eventually, if the CLID is changed, the AP updates the Auxiliary Data as described in clause 6.5.2.3.

6.4.4.8 Channel transition

In any case, smooth transition upon channel switching is ensured, so there is no need to repeat the CL. When the channel is switched, the AP starts acquiring the current event on the new channel. Hence, a new CL is issued, and the device hosting the Sink CPCM Instance will receive it, whether the CL is in-band or out-of-band. The operation depends on the repletion rate of the signalled Usage Rules (see clause 10.1.3).

There is no relationship between channel selection and SAC renewal, so the channel transition is not impacted by SAC establishment.

However, if a SAC renewal was scheduled to happen at the same time as a channel change; the Acquisition Point may preferably send first the new CL through the SAC, using the old SAC session key, and then proceed to the SAC key renewal. Else, this could result in a longer delay before the new channel can be displayed.

6.4.5 C&R regimes

6.4.5.1 Multiple C&R regimes

Upon Acquisition, it is possible to make content available to DVB-CPCM Instances which are compliant with different C&R regimes. Several cases may occur:

- Some or all of the CPCM C&R regimes depend upon the same Root Authority and the rights granted for these C&R regimes are the same. In this case, a single Content Licence may be generated for these C&R regimes. The flag for each authorized C&R regime in the C&R regime is asserted in the Content Licence.

- In other cases, different Content Licences have to be generated for each C&R regime sub-group corresponding to the above description. The different Content Licences are recommended to be transported in-band with the content. It is also possible to exchange the CL out of band using one CPCM message, adding each additional Content Licence (and associated auxiliary data, if any) in a dedicated optional field.

NOTE: In the above cases, some trust needs to exist between the different involved C&R regimes as the CL will be transported under the protection of one of them only.

6.4.5.2 Interoperability between C&R Regimes

When two C&R regimes do not trust each other enough to share content as described in clause 6.4.5.1, it is still possible to enable interoperability using the Export mechanism. In this case, when moving between two C&R regimes, content will have to go through an Export Point of the Source C&R regime and then an Acquisition Point of the Sink C&R regime. These two functional Entities are recommended to be implemented in the same device or even within a single CPCM Instance. Else, mechanisms that are proprietary to the two C&R regimes need to be defined and used for the transmission of the USI and of the content or the CPCM descrambling key.

C&R regimes may agree on a defined mapping that allows automatic translation of the USI during the movement between the two C&R regimes. The mapping rules to go from C&R regime A to C&R regime B may be different from the ones needed to go from C&R regime B to C&R regime A.

The scrambling of the content item may be preserved or may be changed. Changing the scrambling key requires more processing power but allows the decoupling of the security between the two C&R regimes.

6.4.5.3 C&R Regimes and Private Content

Even though DVB CPCM is intended for protecting commercial content, it may also have some applications in protecting personal content. A typical requirement for a personal content protection system would be to allow personal video clips to be viewed by semi-trusted friends and acquaintances, while preventing copying and further distribution of such content. This could be accomplished by utilizing DVB CPCM, and by applying, for example, the usage state "VCPCM, MAD" to the content. This would allow the content to be copied and viewed in the original Authorized Domain but only viewed in other Authorized Domains. Controlling who can have access to the content is beyond the scope of DVB CPCM, but this feature can be implemented at another layer by the content sharing website.

Because the threat model for personal content differs from the threat model associated with commercial content, the compliance and robustness rules may also be different. Parties interested in utilizing DVB CPCM as one component of a personal content protection system are encouraged to set up a dedicated C&R Regime for that purpose.

It is important that this C&R regime takes steps to prevent laundering of unauthorized commercial content.

6.4.6 Content Move

6.4.6.1 Introduction

CPCM has no Content Move Protocol but rather has a Content Licence Move protocol which can be used to achieve the same goal. CPCM content itself may actually be copied freely with no security risk as long as the Content Licence is not also copied. Copy or Movement of Content may be done at any time before, during or after the CL Move protocol. CL Move protocol is the only content management protocol that is a transaction protocol. A bad execution of the protocol could actually lead to the loss of the Content Licence which would disable access the associated Content Item.

EXAMPLE: One of the two devices involved in the Move is faulty, or there is an unstable communications environment. The message carrying the Content Licence is lost but the Content Licence has already been erased from the Source CPCM Instance.

Thus, the Content Move protocol, and the underlying atomic transactions, includes every possible means to prevent any of these adverse consequences from occurring, even when one or both of the communicating devices is faulty or experiencing communications difficulties.

NOTE: The recovery mechanism works only if the two CPCM Instances involved in the Content Move protocol are reconnected together. It is therefore recommended that the recovery information is retained in non volatile storage for as long as possible.

The Content Move Protocol is only executed if allowed by the associated USI, for instance if the Sink CPCM Instance is in the right propagation area.

The protocol is run from Acquisition Points, Storage Entities or Processing Entities to Storage Entities, Processing Entities or Export Points, if the Export system supports this USI. It can never be run with a Consumption Point.

The Content Move Protocol may be done through several CPCM Instances in numerous cases:

- If some Processing is expected to be done on the content.
- If the Move is done towards a Remote Sink Instance and the Source CPCM Instance is not capable of Remote Access: in this case, it may go through the Local Master, if any, or through any other CPCM Instance that is capable of Remote Access. This other CPCM Instance may only be a Processing Entity or a Storage Entity.

In the two above cases, separate and distinct Content Move Protocols will be run separately between each of the involved CPCM Instances. Each individual Content Move has to be allowed by the USI.

The Content Move protocol is used for two USI settings:

- Copy No More content, to Move the single copy of the content.
- VPA content, to avoid starting the Viewing Window before the content is actually copied or viewed.

6.4.6.2 User Interface management

The CPCM specification does not specify the User Interface (UI); it is left to the implementer to deploy a design that is sufficiently user friendly. The design of the Content Licence Move protocol leaves the opportunity for the application to request confirmation from the user once on the server side and twice on the client side. While it is not recommended to actually request confirmation from the user three times, it is recommended to request positive user action at least once on each side, in order to prevent unauthorised remote control of the content. Positive user action includes:

- Launching the protocol on the client device.
- Launching the protocol by requesting the Content Licence from the Sink Instance. This is a positive action on the sink instance only.
- Accepting the protocol to proceed on the server side.

It is strongly recommended to specifically request confirmation from the user for special USI settings allowing the content to be Moved out of the AD (e.g. if the content is marked MLocal. Otherwise, a visiting device in the LAN may be able to 'steal' the single copy of the content without the user noticing it.

6.4.6.3 Normal Behaviour

The Content Licence Move protocol is started by the CPCM Instance from which the Content Licence is to be Moved. After the application has confirmed the Move, a message CL Move Begin is sent to the CPCM Instance to which the Content Licence will be moved.

Upon receiving message CL Move Begin, a CPCM Instance behaves as follows:

- It checks with its Security Control if the Move is allowed.
- It checks with its Application if the Move is allowed.

If both checks are successful, it sends back a message CL Move Ready. Otherwise, it sends a Rollback message.

Upon receiving message CL Move Ready, the source CPCM Instance behaves as follows:

- It checks with its Application if the Move is allowed.
- It requests its Security Control to disable the Content Licence. From that time, the Source CPCM Instance still has a copy of the Content Licence but can no longer access the associated Content Item.
- If both operations succeed, it sends a message Move CL Commit to the other CPCM Instance. Else, it sends a Secured Transaction Rollback message and stops the protocol at this step.

If it receives no replies from the sink CPCM Instance or if it only receives one or more Rollback messages, it stops the protocol there. If it receives both messages Rollback and messages CL Move Ready, messages Rollback are ignored.

After having sent a message CL Move Ready, the CPCM Instance to which the content is about to be Moved behaves as follows:

- If no message is received back, it stops the protocol without informing its security control. Hence, the security control will retain all information about the protocol.
- If it receives a message Secured Transaction Rollback, it informs its security control that the protocol is cancelled. It then stops the protocol.
- If it receives a message CL Move Commit, it requests its security control to proceed to the Move. If the security control confirms the Move has been done, it sends a message CL Move Confirm. At that step, the Instance has a copy of the CL that it cannot use yet. Else, it stops the protocol there.

If a message CL Move Confirm is received by the CPCM Instance from which the content is Moved, this means that the other Instance has successfully received the Content Licence. Hence, it requests its security control to erase the Content Licence and sends back a message CL Move Finish. If the Instance never receives message CL Move Confirm, it stops the protocol there, thus retaining all the information relative to the protocol.

Finally, when the Instance to which the CL has just been Moved receives message CL Move Finish, it informs its security control of the success of the protocol. The CL is then enabled and the protocol stops there. If no such message is received, the protocol stops there without informing the security control.

6.4.6.4 Security Control Behaviour

Security Control modules of each CPCM Instance are involved at various steps of the protocol:

- First, they establish a SAC between both CPCM Instances, or renew it if one already exists.
- Then, the security control of each CPCM Instance stores the CIC identifier of the other Instance. This is done upon sending or receiving message CL Move commit. They also match the roles received in the CL Move Ready or Commit messages with the role present in the certificate.
- They proceed to the actual Move of the Content Licence using messages CL Movement Request and CL Movement Response. CL Movement request is sent by the security control of the Instance to which content is to be Moved. If it receives no response, the protocol stops there.
- Each security control erases the CIC identifier of the other Instance when the protocol is successful. This is done upon sending or receiving message CL Move Finish.

In addition, the security control of the CPCM Instance from which the Content is Moved does the following:

- It disables the Content Licence before message CL Move Commit is sent. Disabling means it keeps a copy of the Content Licence in order to be able to re-send it in case of protocol failure. However, it will no longer be able to use the content. This may be achieved for instance by storing the CL in a dedicated directory.
- It erases the Content Licence when the message Move CL Confirm has been received.

Similarly, the security control of the CPCM Instance to which the content is Moved does the following:

- It stores the Content Licence received in message CL Movement response without enabling it. This may be done by storing it in a dedicated directory.
- It enables the Content Licence once the CL Move Finish is received. The sink CPCM Instance may then access the content.

6.4.7 Content Revocation

CPCM Revocation is a content-based revocation system. This means that a given Instance or AD is not revoked for all Content but only for Content that was Acquired after its actual revocation (and if requested by the Content Owner).

This property is achieved by inserting in the Content Licence the minimum index of the Revocation List a CPCM Instance needs to have. Any list with a higher RL_index contains all the information permitting a CPCM Instance to determine whether it can access a Content Item. There is therefore no need for an Instance to keep track of old Revocation Lists. It needs to keep only to most recent version of Revocation Lists of each C&R regime it implements.

The revocation list verification process as described in TS 102 825-5 [i.14] outputs an RL_index for CPCM certificates that have been revoked. This index corresponds to:

- Index of first revocation for Certificates or AD that have been revoked individually.
- Index of revocation list when the generation of certificate was first revoked, if the revocation results from global revocation of a certificate generation.

Access to content is not permitted if the CPCM Instance does not have a RL whose index is higher than the one requested in the Content Licence. If it does not have the relevant RL, it may get it using RL Acquisition Protocol.

In order to avoid possible delays in getting the required RL, CPCM implementations are recommended to proactively update their Revocation List as soon as they can. This can be achieved using the following ways:

- By inspecting Revocations Lists that may be carried with the content; either in Transport or in File Format.
- By regularly discovering in the home network versions of Revocation Lists known to other CPCM Instances.

6.5 Content Licence Management

6.5.1 Content Licence Generation

Content Licences are always initially generated by an Acquisition Point. Some modifications may later occur in a Storage Entity or a Processing Entity. No modification is permitted by Consumption or Export Points.

6.5.1.1 CL generation upon Content Acquisition

CL creation process is described in TS 102 825-4 [i.9]. The following is also recommended:

- An Instance whose certificate has expired or is known to be revoked (including due to the generation mechanism, see clause 7.8.1.3) does not generate a content licence.
- The Revocation List index is normally set to the value provided by the content delivery system. If no value is provided, the latest value known to the CPCM Instance is used instead. Thus, implementations need to keep the current index value securely, to protect against an attacker trying to erase the latest Revocation List in order to access content from a revoked entity.
- ADID field is set to the ADID of the AD to which the AP belongs. It is recommended to do this even for content that is not required to be bound to an AD (such as MCPCM content). AD protection and identification in the CL allows easier content movement. This does not limit content movement when the content licence is SAC protected. If the AP is not AD aware, this field may be set to zero, or to the CIC identifier of the AP (followed by 0x00 to fill the 9-byte field). Both choices are exactly equivalent. Finally, if the CPCM Instance is not an AD member but is AD aware (i.e. it is a Blank Instance), the following recommendations apply:
  - If the content is marked MCPCM or MLocal, then the ADID may be set to zero. These USI settings allow content to be bound to an AD after Acquisition.
  - If the content is not marked MCPCM or MLocal (i.e. it is strictly AD bound), it is recommended to propose to the user or device application that they create or Join an AD before Acquiring the content into CPCM. If this is not done, even if the Instance later Joins an AD, the content will remain device bound as it will be impossible to rebind the content to that AD.

6.5.1.2 CL Generation in other cases

When a CL is regenerated in a Storage Entity or in a Processing Entity, some fields need to be updated (when some conditions are met) and some others are optionally updated. The list of fields requiring update is as follows:

- Descrambler_information and descrambling_key is updated when the content is Processed and the Processing Entity changes the descrambling key.
- Last CL issuer is updated if the CL is AD protected and not SAC protected, because the use of SAC discards the ability to run the bit bucket scenarios.
- ADID is updated if it was moved to another AD (in which case it is updated to the applicable ADID) or outside of any AD (in which case it is set to zero). Such movement may be done if MLocal or MCPCM is asserted.
- USI are updated when conditions described in TS 102 825-3 [i.7] are met.
- Auxiliary Data digest is updated if any of the auxiliary data fields have changed.
- Changes in auxiliary data are dependent upon extension fields that are carried.

The list of fields that may be optionally updated is the following:

- Content licence protection field value (see clause 6.5.6) can be set to any value authorized by the USI, regardless of the original protection. The CL protection mode is changed accordingly.
- CLID may be updated. In such case, CLID in auxiliary data is also necessarily updated.

A CPCM C&R regime may authorize additional changes or put further constrains on the above ones.

6.5.1.3 Content delineation

CPCM generates a different Content Licence and hence a different CPCM Content item at each change of the Content Licence, for instance following a change in the USI or a change of the Content Scrambling Key. Therefore, if two consecutive content items with different attached usage rules are broadcast, this will automatically result in two different CPCM Content Items.

However, if consecutive broadcast content have the same attached usage rules, CPCM has no means to determine that it needs to generate different CPCM Content Items. It is therefore the role of the delivering CAS to inform the Acquisition Point of the content change so that CPCM may generate different CPCM Content Items. In the absence of such information, CPCM will apply the same scrambling key as long as this is permitted by the applicable C&R regime. This may result in a weakened user experience as one broadcast content item could be split into two CPCM Content Items, each also including a different broadcast content item.

For Free-To-Air content, where no CAS or DRM is available to securely signal to the Acquisition Point the content item change, the same problem exists for two consecutive content items with identical attached rights. In this case several solutions may be defined to inform the Acquisition Point, including but not restricted to the following:

- Changing for a short period of time the attached rights so that CPCM automatically generates a new Content Licence and a new Content Item.
- Using other signalling available within the broadcast content to derive the content change.

EXAMPLE: The content change may be done based upon the EPG information or by a DVB-event-ID change.

6.5.2 Use of Auxiliary Data

6.5.2.1 Generation of Auxiliary Data by AP

When the CPCM AP acquires CPCM Content Item, it generates a CL. In parallel, it can generate a CPCM_auxiliary_data structure to carry data which applies to the Content but is not supported by the CL format. This structure is transported independently from the CL by being embedded into the Content item. It is securely bound to the Content Item by inserting its digest into the CL, which in turn is cryptographically bound to the Content Item and signed. Symmetrically, the CLID of the bound CL is inserted into the CPCM_auxiliary_data structure.

The AP should proceed as follow:

1) It generates the CLID of the CL for the acquired Content Item.
2) It generates the CPCM_auxiliary_data structure with the format defined in TS 102 825-4 [i.9]. The set of inserted Auxiliary Data elements is determined by the intended usage and AP functionalities (e.g. CPCM_rights_issuer_URL, CLC_private_data), the original signalling used to generate the USI (e.g. original_full_USI, original_FTA_control), and may also be requested by the CPCM specification (e.g. CPCM_geographic_area, CPCM_extension_data) or the C&R regime (e.g. other_CPS_export_data).
3) Then the AP puts the CLID into the content_licence_identifier field of the CPCM_auxiliary_data structure.
4) Then it computes the digest of the resulting CPCM_auxiliary_data as described TS 102 825-4 [i.9].
5) Then, it puts this digest into the CPCM_auxiliary_data_digest field of the CL.
6) Eventually, it embeds the CPCM_auxiliary_data structure in the Content item, according to TS 102 825-9 [i.11]. In the case of adaptation to MPEG-2 TS, the AP inserts the structure into an Elementary Stream (ES) of one service which is part of the CPCM Content item, and updates the corresponding PMT with a CP_descriptor indicating the PID of the ES in association with the CP_system_id allocated for Auxiliary Data carriage signalling.

6.5.2.2 Use of Auxiliary Data by sink device

The possible usage of Auxiliary Data by sink devices varies according to the Auxiliary Data elements present in the CPCM_auxiliary_data structure. Sink devices may want to make use of the private data, or can be forced to take them into account by the specification or the C&R Regimes. Devices that use the Auxiliary Data need to be able to retrieve the structure. They should proceed as follow:

1) Check for the presence of the optional CPCM_auxiliary_data_digest field in the CL related to the received Content item.

2) If this digest is found, check for the presence of an Auxiliary Data structure in the Content item that has the CLID of the handled CL. The format and location of the Content item embedded structure is specified in TS 102 825-9 [i.11]. In the case of adaptation to MPEG-2 TS, the device checks the PMT of the service to find the CP_descriptor having the CP_system_id allocated for Auxiliary Data carriage signalling, and extracts the PID of the ES transporting the CPCM_auxiliary_data structure. If no information is available at one step of this process, the sink device should not handle the Content and return an error but this should not happen if the chain of devices that had handled the content is trustable, as underlined in clause 6.5.2.3.

3) If the Auxiliary Data structure is available, the sink device verifies the digest of the retrieved CPCM Auxiliary Data is equal to the one found in the Content License. If the matching fails, it may either not handle the Content and return an error, or process the content anyway. This is determined on a per-case basis by the specification or the C&R regime.

4) If the digest matches, the sink device processes the structure by retrieving the elements that it needs to handle the Content correctly.

6.5.2.3 Maintenance of Auxiliary Data in concordance with CL

Auxiliary Data are bound to the CL through their digest, and conversely by carrying the CLID. Hence, any device issuing or modifying a CL handles the bound Auxiliary Data together.

All fields that are not specified by TS 102 825-4 [i.9] as being modifiable when a CL is re-issued for a given Content item, are kept unchanged from the original CL, unless otherwise permitted by the applicable C&R regime. Since the CPCM_auxiliary_data_digest is not amongst the modifiable fields, it results that Auxiliary Data can be only generated by a CPCM AP at acquisition time and cannot be modified, unless the applicable C&R regime explicitly allows modification of the digest in the CL.

A CPCM instance that has the right to re-issue or modify a CL proceeds as follow:

1) If the new CL has a new CLID:

   a) The instance puts the CLID into the content_licence_identifier field of the CPCM_auxiliary_data structure. For this, such a device needs to have the capability to conform to TS 102 825-9 [i.11] in order to be able to retrieve the Auxiliary Data embedded into the Content item, proceeding as described in clause 6.5.2.2, and to replace it with the modified structure, proceeding as described in clause 6.5.2.1.

   b) Then it computes the digest of the resulting CPCM_auxiliary_data as described in TS 102 825-4 [i.9].

   c) Then, it puts this digest into the CPCM_auxiliary_data_digest field of the new CL.

2) If the new CL keeps the CLID of the original CL:

   a) The instance just copies the CPCM_auxiliary_data_digest found in the original CL into the new CL.

   b) In order to ensure data consistency and avoid later errors; it should verify the presence of a CPCM_auxiliary_data structure embedded into the Content item, proceeding as described in clause 6.5.2.2.

EXAMPLE: A Storing Entity using the DVB-FF specification (TS 102 833 [i.15]) for storing Content should guarantee the presence of Auxiliary Data matching the digest in the dedicated Auxiliary Data box defined by DVB-FF.

6.5.3 CLID Generation

The CPCM System can manage CPCM Content Licences independently of the Content items to which they apply. This gives the benefit of CPCM being more easily able to interoperate across several home networks, but it necessitates that the CPCM System ensures that every CPCM Content Licence is uniquely identifiable. To this end the CPCM Content Licence contains the field Content Licence Identifier (CLID).

The CLID is a 128-bit number (16 bytes) that uniquely identifies the CPCM Content Licence, providing a link between the CPCM Content Licence and the associated CPCM Content item within the CPCM System.

The CLID is generated upon Acquisition of the corresponding CPCM Content item.

It is impossible to build a global registration system for CPCM Content Licence Identifiers, so bearing this in mind, the link to the CPCM Content item needs to be maintained. This is achieved by:

a) the Acquisition Point which generated the CPCM Content Licence allocating the CPCM Content Licence Identifier;

b) by the application of the chosen CLID allocation scheme; and

c) by the corresponding setting of the scheme-specific Content Licence identifier.

The CLID consists of an allocation scheme code (1 byte), and an identifier that is unique within each allocation scheme. The 15 bytes of scheme-specific Content Licence identifier is divided into a pre-allocated part and a part which can be freely allocated by the Acquiring CPCM Instance. The respective sizes of these two parts vary according to the allocation scheme.

A new CLID is allocated whenever a Content Licence (CL) is generated for a new CPCM Content item that is Acquired. The Acquisition Point selects the allocation scheme it wants to apply. Content Licences may also be generated by Storage Entities or Processing Entities. In this case, the same CLID may be re-used or a new one may be generated (see clause 6.4.4.7). The Storage or Processing Entity also selects the allocation scheme it wants to apply.

Each CLID allocation scheme specifies its own method to assign unique Content Licence identifiers within the respective scheme. These are detailed below.

Additional schemes are also proposed for DVB delivery networks (see clause 9.5.1.1).

6.5.3.1 CLID based on CPCM Instance Certificate Identifier Only

In this scheme the pre-allocated part of the scheme-specific Content Licence identifier consists of the CIC Identifier of the Acquiring, Processing or Storing CPCM Instance. The freely allocatable part is the remaining 7 byte (more than 10^{16}).

Using this scheme, allocation will be done independently from the source from which the Acquisition Point acquires the content. This scheme is also recommended for Processing and Storing Entities as it does not interfere with the scheme that is selected by the AP.

6.5.3.2 ISAN

In this scheme the pre-allocated part of the scheme-specific Content Licence identifier consists of the International Standard Audiovisual Number (ISAN) of the Input Content item, and is 12 bytes in length. The freely allocatable part is the remaining 3 bytes (about 16 million) which have to be allocated differently if the same content is Acquired, Stored or Processed several times.

This scheme is dependent upon the Acquired Content Item carrying an ISAN identifier. Not all Acquired content will have this, so implementations can not only rely on ISAN based schemes.

This scheme is thus more difficult to use as the Acquisition Point has to determine whether it already Acquired a Content Item or not and what values of the remaining 3 bytes were already allocated. The same problem exists for Processing and Storing Entities. A counter can be used to that end so that only the latest allocated value needs to be stored.

6.5.3.3 Truncated ISAN

In this scheme the pre-allocated part of the scheme-specific Content Licence identifier consists of the ISAN without the version part and hence is 8 bytes.

EXAMPLE: Different episodes of a TV series have different ISAN with different version numbers but will have the same truncated ISAN.

This scheme is dependent upon the Acquired Content Item carrying an ISAN identifier. Not all Acquired Content will have this, so implementations cannot rely on ISAN based schemes alone.

This scheme is thus more difficult to use as the Acquisition Point has to determine whether it already Acquired a Content Item or not and what values of the remaining 3 bytes were already allocated. The same problem exists for Processing and Storing Entities. A counter can be used to that end so that only the latest allocated value needs to be stored. Thus, there is a small advantage to using truncated ISAN compared with using ISAN as fewer values will need to be recorded.

6.5.4 Content Licence Verification

Content Licence verification is to be performed, as described in TS 102 825-4 [i.9], each time a CL is received or stored. If the verification fails (e.g. because the message carrying the CL was corrupted), the content is not accessible. The CPCM Instance may request the CL again, and, if no valid CL can be obtained, it may try to run the CL Re-Acquisition protocol as described in the next clause. The same applies if Auxiliary Data are expected to be attached to the CL but are not available (i.e. if the CL includes an auxiliary data digest). In this case, the CL cannot be used until the relevant Auxiliary Data are obtained.

6.5.5 Content Licence Re-Acquisition

CPCM includes several means by which a new Content Licence may be re-Acquired in order to get new rights. This can be done using the Content Licence Re-Acquisition Protocol through the CLC that identifies the Acquisition Point that first created the Content Licence or based on a URL that would be present in the Auxiliary Data. These mechanisms and others are further described in TR 102 825-11 [i.8].

The Content Licence Re-Acquisition Protocol may be launched following some user interaction. An alternative for implementers is to propose to run it as soon as the user tries to use the Content in an unauthorized way, either because he never acquired such rights or because corresponding rights have expired.

6.5.6 Content Licence Protection

CPCM supports three modes of Content Licence protection:

- AD Secret only: in this case the entire CL (before descrambling_key encryption) is signed using the MAC algorithm described in TS 102 825-5 [i.14]. Then, the descrambling_key field is encrypted. AD Secret is used for both operations.
- SAC protection only: in this case, when the CL is exchanged between CPCM Instances, the entire message carrying the CL is signed using the MAC algorithm and then encrypted (message header excepted) using the SAC session secrets. The device secret is used for storage protection (both for encryption and authentication).
- AD Secret and SAC protection, in which AD protection is first applied and then SAC protection.

NOTE: SAC protection is temporary and is removed as soon as the message carrying the CL is received. Thus, the CL will be readable as soon as it is received from the security control. AD protection is permanent and is not expected to be removed. The USI and other fields in the CL need to be readable and hence are left in the clear.

The choice of the CL content protection mode depends on the USI setting. Even if only one USI requires SAC protection, then SAC protection needs to be applied. The same stands for AD protection.

When AD protection (with or without SAC protection) is needed, two possible implementations are possible:

- Applying SAC protection and running the AD membership challenge protocol (see clause 6.3.5.8) with the Sink Instance before sending the content licence. This method reduces the overhead when SAC protection is also needed, as SAC protection is applied anyway. This may also be viewed as more secure since the CL remains under SAC or device secret protection and thus will not be endangered in the case of AD Secret leakage. However, this method only works for unicast CL delivery to a CPCM Instance.

- Applying AD protection as described above. If none of the CL USI requires SAC protection, this method enables bit bucket and multicast delivery scenarios. It also avoids the need to run any protocol before sending the CL and thus might provide a better user experience. Conversely, this protection mode cannot be used when content is transferred between two different ADs (where permitted by the assertion of MLocal, VLocal, MCPCM or VCPCM in the USI).

AD protection is required in most cases. The only exceptions are when content is marked MLocal, Vlocal, MCPCM or VCPCM. However, it is still recommended to use AD protection in the latter two cases as it enables additional content management scenarios such as content multicast and bit bucket. For content exchanged in the Local Environment and marked MLocal or VLocal, but not MCPCM or VCPCM, AD protection can be skipped. This is not the case when content movement is permitted Remotely but constrained to the AD.

EXAMPLE: A content item is marked as MAD and MLocal. When moved locally, no AD protection or membership verification needs to be done, whether the Sink Instance is also an AD member or not. When moved to a Remote AD member, AD protection needs to be applied to the CL.

SAC protection is required when the content is copy protected (i.e. not CCNA), when it is marked VPA (which is viewed as copy no more, see clause 6.4.3.3.2), or when it is marked MLocal or VLocal (only if these USI are necessary for the CL exercise). When it is marked MLAD or VLAD, SAC protection is not required as the proximity can be checked using the last_CL_issuer field mechanism (see clause 6.4.3.4.1).

When AD protection is required and content is not transferred between two different ADs, implementations are free to use either of the two options described above. They are not bound to the protection that was chosen when receiving the CL.

When a Content Licence is transferred between two different ADs, the ADID field is cleared and the AD protection is removed before the CL is sent, then the new ADID is inserted and AD protection applied when the CL is received, during the actual transfer, the CL will always be under SAC protection.

7 Security

7.1 Trust Management

Trust is the necessary pre-condition for a CPCM Instance to be able to establish a SAC with another one.

CPCM Instances complying with a given C&R Regime trust each other based on mutual CPCM Instance Certificate verification and, when explicit trust verification is needed, SAC establishment success.

CPCM Instances complying with two different C&R Regimes deriving from the same Root Authority do not necessarily trust each other. Each CPCM Instance has thus to know which other C&R Regimes it trusts as decided by its own C&R Regime. It will check whether it trusts this C&R regime before proceeding with further Certificate Verification or SAC establishment.

CPCM C&R Regimes deriving from different Root Authorities may also trust each other. In this case, CPCM Instances complying with these C&R regimes need to know the Root Authority certificate of the other C&R regime.

Trust decision between C&R regimes is based on security and/or business reasons and thus out of the scope of this multi-part deliverable.

Trust between C&R regimes is not necessarily mutual.

EXAMPLE 1: A FTA dedicated C&R regime may trust CPCM Instances complying with a Pay-TV dedicated C&R regime while the reverse is likely not to be true.

Once trust has been established between two CPCM instances, all CPCM operations are not necessarily permitted.

EXAMPLE 2: A C&R regime may trust another one for Content management purposes but not for AD management purposes. In which cases, CPCM Instances from the second C&R Regime will not be able to Join the AD of CPCM Instances complying with the first C&R regime but will be able to receive Content whose propagation is not AD controlled.

EXAMPLE 3: Even if two CPCM C&R regimes trust each other, content delivered for one C&R regime (i.e. with one bit set in the C&R regime mask in the content licence) will not be available to Instances of the second C&R regime.

7.2 Keys Lifecycle

A CPCM Instance embeds permanently (i.e. since their creation) the following set of keys and sensitive data:

- Root certificate of its Root Authority
- Root certificates of all C&R regimes it trusts and deriving from any other Root Authorities, if applicable
- List of C&R regimes it trusts and associated Root Authorities and C&R regime mask
- Its own certificate and associated chain of parent certificates back to the Root Authority
- Its private key, associated with its certificate and used upon SAC establishment
- Its CPCM Instance Secret, used to protect Content Licences e.g. in Storage Entities

CPCM Instances also embed transient keys:

- AD Secret, if the CPCM Instance is member of an AD. This key is obtained upon AD Joining or Domain Creation and is retained as long as the Instance is a member of that AD.
- SAC Secrets for all CPCM Instances with which a SAC has been established. These keys are obtained upon success of SAC establishment and are retained until one of the parties stops the SAC or until SAC expires.
- Content Scrambling Keys, used to descramble the content. These keys are obtained upon successful verification of a Content Licence and are kept as long as the CPCM Instance acts upon the related Content.

7.3 Keys and Algorithms Protection

The security of any CPCM Instance implementation is governed by the robustness rules and obligations defined by a C&R regime. However, in anticipation of such requirements, it is likely that C&R regimes will establish rules that will require the implementations to be protected against hardware and software attacks.

Sensitive data include:

- Permanent device secret and private keys
- Root public key and CPCM Instance Certificate
- Transient keys
- Random values generated for SAC establishment
- Counters
- Obfuscating data or methods
- Content
- Software or Firmware
- USI

A possible way to secure sensitive data while maintaining quick access to it is to store such data in non-volatile memory protected by the device permanent secret.

Possible attacks include:

- Side-channel attacks which consist in monitoring parameters (e.g. elapsed time, power consumption…) that are external to the system when an algorithm is running. This observation may actually reveal secret data or secret temporary values (e.g. the seed in a PRNG).
- Bugs exploitation (e.g. triggering a buffer overflow may reveal some secret data).
- Code modifications (for software implementation).
- Fault injection attacks.
- Others not identified here.

7.4 Secure Authenticated Channel

In the process of establishing a Secure Authenticated Channel between two CPCM instances, two types of keys are computed:

- The long-term key K_{perm} is used in trust establishment. For a given pair of CPCM Instances, K_{perm} will always be the same. Different options regarding the handling of K_{perm} include the following:
  - Destruction of K_{perm} immediately at the end of the SAC establishment process: In this case, K_{perm} will be re-computed upon each session key renewal. This is the option which saves the most memory resources but is the least efficient because of the time needed to compute K_{perm}.
  - Save K_{perm} in protected permanent memory: K_{perm} does not need to be re-computed as long as it is not erased from permanent memory. At each session key renewal the certificate revocation status is verified. If the certificate is revoked, then K_{perm} is not used and it should be erased from permanent memory. As the CPCM Instance may establish a SAC with several other CPCM instances, at least the certificate serial number, and possibly the whole certificate, should be recorded with K_{perm}. Otherwise the CPCM instance does not know which K_{perm} to use. The recording of K_{perm} in permanent memory should be at least as secure as other permanent secrets.
  - Save K_{perm} in volatile memory: it is not necessary to re-compute K_{perm} until the next device power-off. If the CPCM instance concurrently handles different SACs with different CPCM instances, this option may require a large amount of volatile memory. As a minimum, the certificate serial number needs to be saved together with K_{perm} in order to be able to handle concurrent SACs. The certificate revocation status needs to be verified at each key renewal. When chosen, this option needs to be very secure against software attacks attempting to access the volatile memory.
- Short-term session keys K_{sess} are used to protect communication between CPCM Instances. These keys need to be saved in volatile memory and are re-computed after each device power-off. A CPCM C&R Regime will define a session key renewal policy based upon session key usage and/or time considerations. The session key renewal protocol is the same as the SAC establishment protocol. The following considerations should be taken into account when implementing this CPCM feature:
  - Session Key renewal should preferably be done in advance of the session key expiration in order to prevent disruption of service
  - Session Key renewal is mandatory before performing AD Join, AD Leave, DC Transfer, Split, Merge or Rebalance protocols.
  - In order to counter some cryptographic attacks a certain key usage and/or a time period as defined by a CPCM C&R Regime should be respected before the CPCM instance accepts session key renewal.

- A SAC can be established with another CPCM Instance in advance of actual usage, as soon as that Instance has been discovered. This allows the speeding up of any operation that has SAC as a prerequisite (e.g. content licence exchange for some USI settings or proximity checking) but requires more memory available to store session keys, corresponding CPCM Instance identifiers and SAC expiry time. Alternatively, SAC may be established only when it is needed and the corresponding SAC key will be erased at the next SAC establishment. This allows for decreasing memory requirements but could cause delays in user experience.

- If supported, it is recommended that implementations attempt to renew the SAC before it actually expires.

7.5 Local Scrambler Algorithm (LSA)

AV data is carried in packets comprised of a clear (never scrambled) part and a scrambled payload containing the compressed audio video data. The clear part (a.k.a. Header) contains information necessary for management of the packets needed for packet filtering and routing, e.g. the PID field in MPEG2-TS header.

Must Stay Clear (MSC) refers to the clear part of the packet. The fact that this data is kept in the clear does not necessarily mean that the scrambling is made independently of it: this data may be used in the generation of the IV, both for CBC and RCBC modes. This is what is done in MDD, MSC Data Dependent, mode. On the other hand, in Must Stay Clear Data Independent (MDI), the IV is computed independently from MSC data.

MDD mode can be used when the following is needed:

- to protect the clear Header part of the content against unauthorized changes; and

- to provide more entropy for the IV generation.

Thus, the MDD-scrambled packets inherently write-protect their clear Header. Using MDD mode, such unauthorized modification of the Header will result in failure to descramble the content. Thus, these changes can be assured to be made in an authorized manner, i.e., only by an entity that knows the descrambling key.

For TS content, MSC data are the four bytes of TS-header plus (optional) Adaptation Field (AF) of the TS packet; both referred to as Header. The transport_error_indicator bit in the TS header is excluded and will not impact descrambling when changed in transit.

MDI mode is typically employed when it is commercially desired to have re-multiplexing of scrambled streams or changes to the MSC can be done on the scrambled packet, i.e., the packet does not have to be descrambled prior to changing MSC (PID or other MSC fields).

MDD mode prevents a number of operations including the re-multiplexing of scrambled packets, the splicing and insertion of scrambled packets, and changing certain fields like PCR in scrambled packets. MDD can be used for instances when the following objectives are to be met:

1) Applications, like games or gambling where it is important to preclude unauthorized PCR changes of a scrambled packet.

2) To prevent unauthorized PID modifications of scrambled packet and to preclude:

   - re-muxing of multiple SPTSs streams to an MPTS;

   - editing of multi streams - splicing of scrambled streams.

3) Integrity protection of the AF as it may contain the service provider's private data needed for processing the stream.

LSA scrambler includes, in addition, two chaining modes. Both can be used in similar applications.

7.6 Time Management

There are two sorts of CPCM Instances that are secure absolute time capable:

- CPCM Instances implementing a secure clock and is allowed to generate secure time autonomously.

- CPCM Instances able to maintain secure time for a given duration, which is determined by the applicable CPCM C&R Regime, once they have obtained secure time using the secure time protocol.

Both sorts are capable to deliver secure time to another CPCM Instance using the secure time protocol.

Secure Absolute time implementation is not a requirement for all CPCM Instances. CPCM Instances that do not implement secure absolute time may obtain it using secure absolute time protocol. To that end, it uses CPCM Discovery Protocol and checks the value of the returned absolute_time_capable in the instance_CPCM_status field. It then runs the Secure Time management protocol described in TS 102 825-4 [i.9].

Secure relative time implementation is a requirement for all CPCM Instances. It can be implemented using a secure counter incremented on a regular basis. This counter resides in volatile memory. The counter bit size is up to the implementer's choice and determines the maximum relative time that can be measured. If it is shorter than SAC session key expiry time or than proximity test persistence, as determined by a C&R regime, the CPCM Instance will have to renew the SAC or to re-check proximity before the counter reaches its maximal value.

7.7 Random Number Generation

The CPCM specification makes use of random numbers. They are used for instance in the SAC establishment process, for key generation purposes (AD Secret generation or Content Descrambling Key in an AP or a PE), for proximity testing (e.g. SRTT test) or for some challenge / response protocols (e.g. AD membership protocol).

Thus, CPCM system security relies partly on the quality of the random number generator. As indicated in TR 102 825-13 [i.5], a C&R regime is expected to define compliance and robustness criteria for random number generation. This clause aims at providing good practises that will help the implementer in meeting the C&R regime requirements.

There are two kinds of Random Number Generators:

- Pseudo Random Number Generators (PRNG) which always generate the same random bit stream from the same seed. PRNGs have been widely studied and are generally inexpensive to implement.

- True Random Number Generators (TRNG) whose output cannot be repeated. TRNG are much more difficult to implement as numerous external factors (power supply, temperature …) can have a large impact on their output. Good TRNG are usually very slow as they need to gather enough entropy before outputting a new bit.

The CPCM specification needs a True Random Number Generator (i.e. a random number generator whose output cannot be repeated). To reduce the implementation costs, it is advised to implement a TRNG only for the purpose of generating seeds that will be used as inputs to a PRNG.

In order to implement a good TRNG, implementers have to find a good source of entropy which may be dependant in part of the nature of the device that embeds the CPCM Instance or is comprised in a separate circuit. This entropy should be minimally sensitive to external conditions or parameters so that the entropy cannot be altered. More guidelines and testing procedures for TRNGs can be found in [i.16].

PRNG have been much wider studied than TRNG. PRNG implementation needs to be secured like any other implementation of cryptographic algorithm (see 7.1). Implementers can refer to [i.17], [i.18], [i.19] or [i.20] to find examples of good PRNG and examples of randomness tests.

A good security practise is to use two different PRNGs (or even TRNGs) depending on whether the output of the RNG will be made public (e.g. for challenge / response purposes) or not (e.g. in Diffie-Hellman Algorithm or for key generation purposes).

NOTE: There is no PRNG specification as it will not provide any value with respect to interoperability and is strictly for CPCM Instance internal usage. To the contrary, specifying one PRNG could endanger many implementations in case this PRNG would be flawed.

7.8 Certificate Management and Revocation

7.8.1 Certificate Management

7.8.1.1 Root Authority

The Root Authority has a public key which is referred as the root certificate in this multi-part deliverable. The root certificate is not signed and has not the structure described in TS 102 825-5 [i.14] and only consists in the Root Authority public key. The root certificate is to be stored in write-protected memory (e.g. in ROM).

7.8.1.2 Certificate chain Verification

Certificate chain verification procedure consists of the verification of all the parent certificates in the chain to or from the root certificate; i.e. it can be done starting with the Root Certificate down to the Instance Certificate or in the reverse order. Parent certificate verification consists of three steps, namely revocation status verification, checking that the field is_signer is asserted and verification of its parent certificate.

This verification can be speeded up by first verifying whether one parent certificate matches one of its own parent certificates. In such case, if the CPCM Instance is not itself revoked (see clause 7.8.2), the parent certificate and all of its parents can be considered as verified.

Alternatively, an implementation may maintain a list of some parent certificates it recently successfully verified. In this case, the implementation verifies that the revocation status of each certificate has not changed at each revocation list update.

7.8.1.3 Aggregation rules

When a certificate chain is verified, values of certain fields in the whole certificate chain may be superseded by the value of corresponding field in a particular certificate in the chain. This feature is named aggregation rule. It allows a CPCM C&R regime to update the value of the corresponding field in all child or parent CPCM certificates by only updating one parent CPCM certificate.

7.8.1.4 Index and Time Expiration Certificate

CPCM supports two kinds of certificates expiry:

- Time-based expiry: an absolute-time may be included as an expiry date in the certificate. This expiry date is to be verified for time-related USI, i.e. in the following cases:
  - If View Window is Asserted.
  - If View Period is Asserted and a first copy or consumption is requested, as the CPCM Instance has to compute a View Window.
  - If content is marked MLAD, VLAD, MGAD or VGAD with a remote access rule different than remote_access_record and if the CPCM Instance wants to assess whether the Proximity or Geographical restriction can be relaxed.
  - If the expiry date has to be verified and the verifying CPCM Instance is not absolute time capable, the CPCM Instance has first to obtain Secure Absolute time. The expiry date has to be both verified by the source and the sink CPCM Instance.
  - A C&R regime may impose additional circumstances in which certificate expiry date has to be verified. It has however to be noted that this verification will never be imposed in the case where a SAC is established for the purpose of exchanging secure absolute time, as secure absolute time would be needed beforehand to verify the certificate expiry date.

- Index-based expiry: an index is systematically inserted in the certificate. The incrementing policy of this index is at the discretion of the issuing C&R regime, but with a maximum value of 255. The C&R regime also specifies, in the Revocation List it issues, a minimal index that a certificate needs to have to be viewed as valid. Thus, by incrementing this index in the Revocation List, the C&R regime may expire a whole generation of certificates. The index can thus be viewed as a way for the C&R regime to expire certificates after a given event has occurred, whereas time-based expiry is automatic.
  As part of the Revocation List verification process, the index validity of the certificate has to be verified each time a certificate is verified.

7.8.1.5 AAA Certification

AAA certification is not defined in the CPCM specification.

There are two kinds of AAA:

- AAA whose trust is granted by a CPCM C&R regime. This is the ADM AAA or the Proximity Control AAA, in case PTAAA proximity tool is supported. In this case, C&R regime delivers to the ADM AAA a CPCM certificate that will enable the ADM AAA to set up a SAC with any CPCM Instance. Each CPCM Instance is to be able to recognize this certificate as an ADM AAA certificate; see below.

- AAA whose trust is granted by the content delivery system or the content owner. In such a case, this AAA may only be contacted through an agent, such as a CA system or a web server, implementing both CPCM and the AAA security system. The security or the certification of this AAA is thus out of CPCM responsibility.

There are several possibilities for a C&R regime to allow any CPCM Instance to identify a given CPCM certificate as a C&R regime certificate:

- Allocating a dedicated extension to the ADM AAA certificate.

- Defining a specific extension during the SAC establishment protocol.

- Allocating a specific unused (i.e. in a normal CPCM Instance certificate) setting to selected fields in the ADM AAA certificate (e.g. LM not capable but DC capable).

- Allocating a specific Certificate identifier or range of identifiers. This method can be further extended by setting the bit is_signer in the certificate fields and by specifying that any child certificate of an ADM AAA certificate is also an ADM AAA certificate. This is a C&R regime option.

ADM AAA certificates may in any case be incorporated in a revocation list.

7.8.2 Revocation

7.8.2.1 General Principles

Revocation criteria are defined by a CPCM C&R regime.

A content provider is expected to require the latest CRL for its premium content but may require older indices for less valuable content.

FTA content providers have a mechanism to disregard revocation (through the do_not_apply_revocation bit). If this mechanism is used, all revocation lists will be disregarded for handling FTA content. This does not apply for operations that would have a wider scope than FTA content, such as AD bound operations.

7.8.2.2 Certificate Revocation

Revocation of a CPCM certificate is decided by the C&R regime with which the relevant CPCM Instance complies or by another C&R regime that trusts certificates issued by the original C&R regime to which the CPCM Instance belongs.

Certificate revocation is done on a per C&R regime basis, i.e., a given certificate may be revoked by one C&R regime but can still be trusted by another one.

Certificate revocation is always done with regards to a CRL index: this means that certificate identifiers present in a CRL are viewed as revoked for all contents requiring a CRL with a higher index. Thus, content that was Acquired into CPCM before the certificate revocation will still be available, as it was necessarily issued with a CRL index in the CRL lower than the most recent one at the Acquisition time. This regards only operations linked to a given CPCM Content Item such as SAC establishment or proximity check in order to the consumption or the copy of a given Content Item. No operations that could impact Content Items for which the CPCM Instance are actually revoked, such as AD management, can be performed once a certificate identifier is included in a CRL.

7.8.2.3 Tree Structure

The tree structure enables a CPCM C&R regime to revoke a whole family of certificates while embedding only one certificate in the list. Such a mechanism is not available for AD revocation for which only revocation based on the AD Secret digest, and thus on a per Authorized Domain basis, is enabled.

Tree structure based revocation should be used very carefully as, per definition; it necessarily impacts uncompromised implementations and thus genuine users. It is thus advised to not use this mechanism if there are no easy means to update a CPCM Instance implementation (e.g. by software downloading).

Revocation can be made to any level up to the C&R regime level. A C&R regime is not likely to issue a CRL that revokes its own master certificate. However, Content can be preserved from all the certificates issued by a given C&R regime by not asserting the corresponding bit in the C&R regime mask.

7.8.2.4 AD Revocation

Revocation of an AD is decided by a C&R regime for which AD related operations were permitted.

AD revocation is done on a per C&R regime basis, i.e. a given AD may be revoked by one C&R regime but may still be trusted by another one.

AD revocation is always done with regards to a CRL index: this means that an AD Secret digest present in a CRL is viewed as revoked for all contents requiring a CRL with a greater index. Thus, content that was Acquired into CPCM before the AD revocation will still be available (as it was necessarily issued with a CRL index in the CRL lower than the most recent one at the Acquisition time).This regards only operations linked to a given CPCM Content Item (e.g. SAC establishment or proximity check in order to the consumption or the copy of a given content item). No operations that could impact content items for which the AD is actually revoked (e.g. AD Management) can be performed once an AD Secret digest is included in a CRL.

The AD Secret digest has been given preference in the specification over the ADID for inclusion in the CRL as hackers are very likely to publish the AD Secret when breaking an Authorized Domain.

When an AD member detects that its current AD has been revoked, it remains a member of the revoked AD, so as to be able to consume Content that was Acquired before the AD revocation. However, it will respond to certain AD related requests, such as LM Election or AD Join, with an 'AD is revoked' error message. The user may force the Instance to Leave the AD, but the normal protocol may not be used. Instead, the ADID and the AD Secret will be erased from the Leaving Instance that will become a Blank Instance, without running the usual Leave protocol. Leaving the AD in this manner will prevent further access to AD-bound content, so the user should be clearly warned about the consequences and asked for confirmation.

NOTE: The AD Leave protocol is used to ensure the ADSE counts are correctly updated. This is unnecessary for a revoked AD, as no further AD Join will be permitted in any case.

AD Join, DC Transfer, Split, Merge and Rebalance and LM Election protocols cannot be run in a revoked AD. If initiated by an Instance that is unaware of the revocation, an 'AD is revoked" error will be generated. However, receiving such an error message does not cause the Instance to see the AD as revoked. It may only happen if the corresponding Revocation List is received. Therefore, it may be appropriate for an Instance to request the latest Revocation List if it encounters this error.

7.8.2.5 CRL delivery and storage

CPCM revocation mechanism is based on the fact that content providers are expected to require the latest CRL for their content. Thus, if the user consumes premium content, he will get better user experience if he already has the latest version of the CRL: It is therefore recommended that an implementation should get the latest CRL version as soon as possible in order not to delay the user experience. This can be achieved by using the RL Acquisition Protocol anytime or upon CPCM discovery protocol. Content cannot be accessed if the CPCM Instance does not have a CRL with an index equal to or greater than the index present in the Content Licence.

When a CRL verification is required for purposes other than content access, CPCM Instances use the latest CRL version they have available. An RL Acquisition protocol may be run beforehand to ensure that the latest CRL has been obtained.

CPCM Instances do not need to store more than one CRL issued by a same C&R regime as the latest CRL includes all the information needed by a CPCM Instance upon CRL verification: it includes the index of first revocation for all revoked entities which allows a CPCM Instance to determine whether a CIC or an AD was already revoked at the Acquisition time of a given content.

Replacement of a CRL by a more recent one can be done pro-actively: CRLs may be transported together with the content and a CPCM Instance may look for recent CRLs even within a content it does not try to access.

7.8.2.6 Revocation and Versioning

CPCM Revocation Lists are issued for one CPCM version. Therefore, C&R regimes need to maintain and issue Revocations Lists for all existing versions of CPCM.

If a C&R regime wants to revoke all the implementations of one given version of CPCM, possibly because there was a class attack on that version, it can do this in several ways:

- It can revoke high level certificates in the hierarchy so that all CPCM Instances under that hierarchy are also revoked. In such a case a different hierarchy will have to be used for the new CPCM version.
- It can request, in CPCM Content Licences corresponding to that version, an RL_index that is higher that the index of the latest issued Revocation List. As the corresponding RL does not exist, no CPCM implementation from that version will be authorized to access to the Content.

8 ADM implementation guidelines

8.1 Introduction

This guidelines aim at helping the implementer understanding and implementing ADM protocols as well as describing how advanced protocols, such as a single Authorised Domain spanning upon CPCM and another DRM system. Description of standard ADM protocols can be found in TS 102 825-7 [i.12].

8.2 Authorized Domain Management

8.2.1 Implementing ADM Functionality

The Authorised Domain Management functionality can be implemented in many different ways. One approach is to develop a separate subsystem within a CPCM implementation. ADM is deliberately more or less separate from most other CPCM functionality, which would allow development by a separate team of developers if appropriate. The ADM specification could also be used for non-CPCM applications that require domain management.

The ADM functionality needs to be implemented with sufficient robustness to meet the requirements of the governing C&R regime, however the design of the ADM specification allows it to be used outside a secure computing environment (e.g. in software on a general purpose CPU), as all the necessary handling of secret information is delegated to the CPCM Security Control functionality.

8.2.1.1 Multi-threading of ADM

The ADM specification is written in terms of a single-threaded process. Implementers may wish to consider implementing ADM using a multi-threaded approach to allow parallel processing of ADM messages. While ADM transactions do not occur very frequently, implementers should consider whether their solution is likely to encounter multiple simultaneous ADM events, and give consideration to the user experience should this occur. The most likely case for multiple events is probably during power-on or reconnection of multiple CPCM-enabled devices at the same time, for example after a power cut event.

8.2.1.2 Multiple ADM Instances within a Device

It is permissible to have multiple CPCM instances within a single device. While a given CPCM instance can only be a member of a single domain, it is permissible for the ADM functions of different instances to be members of different domains. This allows a single device to participate in multiple ADs, though any domain-bound content item will be bound to a single AD and hence to a single CPCM Instance.

A specific case of this is described in TR 102 825-11 [i.8] for a Hosted CPCM Service, where a network based server supports multiple consumer home networks. In such an implementation, the server will have a database of ADs, one per consumer household, with each "row" of the database holding the ADM information for each household. Thus the CPCM Instance is implemented as a row in a database table, rather than as a discrete block of software. This is comparable to a banking service, where a customer's account is implemented as a data structure alongside many other such accounts. Nobody worries about money slipping from their account to someone else's account. In the same way, CPCM Content bound to an AD stored in this way will also not leak to another consumer's domain.

Another situation that can occur is where there are multiple implementations of CPCM running in a single physical device.

EXAMPLE: A personal computer runs an Internet television application which implements the CPCM standard to allow content to be shared with other devices in the home. Some months later, another application from a different vendor is added to the PC. This also implements CPCM. The two applications are unaware of each other, and are able to join separate domains.

8.2.2 Local Master and Domain Controller

Authorized Domain Management protocols are run thanks to the presence of two roles in the home network:

- A Local Master which is in charge of networking aspects, such as discovery, remote messaging. It will be the CPCM Instance to which Local CPCM Instances will address all ADM related requests. If it is not also Domain Controller, its role is then to forward the requests to the relevant Domain Controller. This process is called delegation.

- A Domain Controller which is in charge of enforcing Authorized Domain Size and Extent (ADSE) rules. Any operation affecting the ADSE resources can be achieved only if the DC is involved. The Domain Controller provides a central point of management of the domain, keeps track of the current domain size and other ADSE parameters, and greatly aids in constraining it to a single household as required by C&R. It was originally envisaged to use a fully distributed domain management function, but this proved too complex to justify. The CPCM specification allows for the following operations on the DC:

  - DC transfer which allows a user to select which CPCM Instance will be its DC, see clause 8.3.3.6.

  - DC splitting which allows a user to have a DC in each of the Local clusters of its AD, see clause 8.3.3.7.

  - DC merging which allows a user to re-combine split DCs into a single DC.

  - DC rebalancing which allows a user to transfer ADSE counts (and hence permissions) from one DC to another. This is for instance useful if one split DC has reached or is approaching a ceiling while another DC of the same domain has more availability.

A Domain Controller is generally also the Local Master for its cluster. This avoids the need for the delegation mechanism, where most of the Instances have the ability to communicate directly with each other. If several DCs are present in the same cluster, the one which has the most available capabilities is selected (see also clause 8.3.2.7).

If no DC is available locally, the Local Master is responsible for redirecting the AD related messages to a remote DC. As several may exist, the Local Master should select the one for which the requested operation can be permitted. Alternatively, the LM may try to run the protocol successively with several DCs until the request is accepted.

EXAMPLE: if the Local Master forwards an AD Join request, there is no need to transfer that request to a Remote DC controller which has reached its ceiling for Remote Joining.

While the number of connected Domain Controllers to a given LAN is not controlled (it can go from zero, if no DC is currently connected to the LAN to several if the DC was split and several of the DCs are currently connected), it is expected to always have one Local Master for a LAN. This is achieved using the Local Master Election process (see clause 8.3.2.7) which elects one LM amongst the LM capable Instances of the AD. A Local Master may re-launch the LM election process before being disconnected from the network or just leave the local cluster without any LM. Any AD member will be able to launch a new LM Election when it will need the intervention of a LM.

NOTE: It might also happen that two LM exist simultaneously on a given LAN, e.g. because a network connection has been re-established between two parts of the LAN that had each a LM. This should not cause any user experience problem as each AD member will continue to use the LM it knows. It is recommended to re-launch a new LM election process as soon is the presence of two LM is detected.

8.2.2.1 Local Master role

CPCM was designed from the outset to support cases where the Authorised Domain extends beyond the physical boundary of a single dwelling. It was considered that holiday-makers and business travellers are quite likely to carry more than one entertainment device with them, and will expect to be able to move and play content between these devices even when away from home.

So, the Local Master concept was introduced to ensure that a "cluster" of CPCM devices in communication with each other in a given location are able to focus their communications with the "home base", rather than each and every device having to be able to reach the home directly all the time.

The Local Master approach allows most devices to ignore their location with respect to the active domain controller, as they only need to talk to the Local Master. They can discover the Local Master easily, so they do not need to "phone home" themselves. The Local Master is elected automatically, so that the user does not usually need to worry about which devices are currently connected or powered on. Also, the Local Master (the LM), provides CPCM with a means to ensure that a new device being added to a domain is in close proximity to an existing domain member which greatly reduces the threat of unauthorised devices joining a consumer's AD without their knowledge or consent.

8.2.2.2 DC controller Splitting Capability

There are two main reasons for allowing a DC to split. Firstly, it is recognised that a single Domain Controller function becomes a single point of failure for the whole domain. Should the Domain Controller device fail completely (or be stolen or otherwise lost), it would be impossible to add new devices to the domain. Over time, as other devices fail or are replaced the domain would shrink and eventually the content bound to that domain would become unusable. Allowing the Domain Controller function to be split means that the loss of one such device does not result in total loss, and remedial action such as asking the C&R Regime for permission to increase the available counts to accommodate those lost becomes a possibility.

Secondly, there are valid use cases such as when a household owns a second property or vehicle such as a mountain cabin, boat or caravan which does not have regular Internet connectivity. In such a case, a "secondary" Domain Controller with autonomous authority for some of the domain becomes a useful facility. Devices can join the domain in either location, and can be moved between the two (with the content intact) and function fully in both environments. This avoids setting up network connections between the different clusters for DC transactions. The drawback is that the ADSE counts (see clause 8.2.7.1) will be split between the different DCs. This could lead to situations where the user cannot perform an AD operation in one location that would have been possible in another. This problem can be mitigated using the DC rebalancing mechanism when connectivity allows.

8.2.2.3 DC Transfer capability

The default DC is the CPCM Instance that created the AD, which may not be the best choice for the user.

The DC transfer capability allows the logical DC function to be moved from a device in a managed way, so that the CPCM Instance that was originally hosting the Domain Controller function can be temporarily or permanently removed from the device or the network, perhaps so that it can be sold or otherwise decommissioned, without adversely affecting the operation of the AD. It may also be useful to do this if a new device with increased DC functionality or reliability is added to the network, as it can do the job better than the original DC device. For instance, it is not desirable that a portable device plays the role of the DC since it might be away when a DC intervention is expected.

8.2.3 Blank Instances

There are different implementation options for a CPCM Instance to decide that AD management should commence:

- As soon as the device is first switched on
- As soon as content which is bound to an AD is received. At this time it is possible to discover CPCM Instances that are members of the indicated AD and try to Join that AD.
- Only on user request.

The first option is recommended for an AP that is likely to Acquire content that needs to be bound to an AD.

The advantage of the first option is that the implementation will not delay the consumption of the first AD-bound content. For devices in a second home, for instance, Domain Controller might not be available at that time. A drawback is that in the absence of network connectivity when the device is first switched on, a new AD could be created although the household already has an AD elsewhere. It is therefore recommended that implementations clearly request the user to confirm this choice before proceeding to AD creation.

The advantages of the second option are that it allows postponing the selection of the AD where the device is to be installed. It also allows the user to enjoy more quickly content that is not AD bound. This option also allows making AD management more transparent for the user as they will not have to select an AD from a list of available ADs like in the case where several ADs are discovered; as can occur under the first option.

The third option should be avoided except for technology aware users.

The same options exist for a device that has just left a domain: it may decide to immediately create a new domain or to stay blank until either the user triggers the domain joining process or it receives AD-bound content.

NOTE: It is not always necessary to join an AD before handling AD-bound content. For example, if the content is also marked as MLocal or VLocal, and the content is stored Locally, content can be obtained using SAC protection. Also, content that is currently protected using the AD Secret may be obtained if it is marked as MCPCM or VCPCM even when it is not Local, though this will require the source to generate a new Content Licence under SAC protection.

8.2.4 AD Lock

When a user wishes to install a device on an AD, a possible protocol is to first discover the available ADs and then to select the desired AD on the new device. Consequently, it is theoretically possible to install a device on Alice's domain without Alice noticing it. This potentially leads to problems for Alice, such as privacy issues, restricted ability to install new devices, and other problems.

The Join procedure leaves room for user confirmation at many points. If it is not desired to actually request confirmation from the user at each point, it is strongly recommended that the implementation requires user confirmation at least once on the Joining Device and on the DC or the LM.

It is therefore advised to implementers to give the user the possibility to lock its Authorized Domain. When locked, a domain controller will ignore Join requests. The detailed specifications and features of this AD lock are left to the implementer.

In addition, implementations of Domain Controllers should require the use of a password, PIN, smart card, fingerprint, or similar mechanism to authenticate confirmations of joining the domain. A simple dialog box or button may not be sufficient, as there remains the risk of a visitor to the household surreptitiously joining their equipment to the domain without the permission of the household. Acceptance of a joining device is still subject to ADSE verification.

8.2.5 Non Self-Managing Devices

AD Management includes support for devices that have a very basic UI. This allows such a device to benefit from other device UI to securely join a selected domain (as the selection cannot easily be done using the simple device's own UI).

EXAMPLE: A set of audio speakers are enabled with CPCM. The user interface is limited to a single push button and a flashing light. The user requests the AD Join by pushing the button. When a confirmation is required, the light flashes and the user pushes the button again. When the AD Join is successful, the light glows steadily or changes its colour. If the user wishes the device to leave the AD, they push the button again.

To provide a satisfactory user experience, implementers of such devices need to provide the user with sufficient information allowing him to distinguish between the following ADM states:

- Proposal for AD joining.
- Error state.
- And if the device the DC capable:
  - Proposal for AD creation.
  - Authorization of joining for another device (in the case where the device is the domain controller).

NOTE: It is strongly discouraged to implement a DC capable CPCM Instance on a device that has a very limited UI. However, this discouragement would not be applicable in case of an appliance without a screen if it offered a richer remote UI such as a scripted web page that the consumer could access from a browser.

Additionally, the user needs to be able to accept or refuse each of the above proposed actions or to leave the error state (e.g. if no action is performed during one minute).

8.2.6 Authorized Domain Capabilities

There are several options for an AD aware CPCM Instance to implement ADM capabilities, each of which involves different functionalities that need to be implemented.

8.2.6.1 No AD capability

CPCM Instances are not required to implement any Authorized Domain-related function. However, a CPCM Instance that is not even AD aware would be very restricted in its ability to handle or Consume content; such a CPCM Instance would actually only be able to exchange content which is marked as 'MLocal', 'VLocal', 'MCPCM' or 'VCPCM' with other CPCM Instances. The propagation of all other content is actually only authorized within a given Authorized Domain and thus cannot obtained by or transferred from a CPCM Instance that is not AD aware. Many services will allow greater flexibility of content within an AD, including some FTA broadcasts, and many of the benefits of CPCM derive from the use of the AD, therefore a product lacking AD capability will be much less attractive for the consumer.

NOTE 1: A CPCM Instance that is not AD aware and that contains both an Acquisition Point and a Consumption Point (or an Export Point) can Acquire and Consume (or Export) content that is marked as MAD, VAD, MGAD, VGAD, MLAD or VLAD as it is inherently compliant within the device. If it is also a Storage Entity or a Processing Entity, it can also Store or Process the Content, however this content item can only be Consumed or Exported within the same device and never transferred under CPCM protection to another device.

NOTE 2: CPCM Instances that are not AD aware are not considered as Blank Instances for ADM protocols. They are just ignored.

8.2.6.2 AD awareness

An AD aware CPCM Instance that is not ADM capable is a CPCM Instance that is able to know the AD secret and to enforce AD related USI. However, it does not use ADM protocols to get the AD secret but uses instead proprietary methods, as allowed by the applicable C&R regime.

EXAMPLE: A smart card implementing part of an AP may get the AD secret using proprietary protocols of the CA or DRM implemented by the AP while the rest of the AP will be ADM capable.

If the standard ADSE method is used, an AD aware CPCM Instance has to maintain the history_count and the history_ceiling. It also has to be able to store, though not necessarily securely, the latest version of the AD internal record. This storage is not done for security purposes, but to increase the chances of having an up-to-date version of the record, so that an ADMAAA may re-establish the AD according to the stored version in case of a serious problem.

While it is permitted to implement a CPCM Instance that is AD aware only, for home network connected devices it is better to make them ADM capable.

NOTE: A CPCM Instance which is only AD aware will always be treated as Blank by ADM protocols until its AD membership is established by proprietary means. Its membership will continue unaffected by ADM protocols, unless proprietary means are used to change this.

8.2.6.3 ADM Capable

ADM capable Instances are able to run the protocols described in 8.3 at least as a Blank Instance or as a simple AD member.

They are AD aware so they have to implement the corresponding functions.

In addition, they have to implement protocols described in clause 8.6.3.1.

This is the minimum level of AD capability recommended for most CPCM implementations.

8.2.6.4 Local Master capable

CPCM devices are not required to be Local Master capable. However, implementers are recommended to include Local Master functionality in all devices that have sufficient resources to provide this function. Local Master capable devices are necessarily ADM capable and thus have to implement the corresponding functions.

For Local Master functionality, there should be some means provided (beyond the scope of CPCM specification) to configure the device with the network address and Device ID of the appropriate Domain Controller. This is especially important when the Domain Controller will be Remote from the local home network, and thus will not be easily discoverable by other means.

In addition, LM capable CPCM Instances have to implement the LM election protocol and be able to determine their LM capability for that election.

Other protocols required for LM capable devices are given in clause 8.6.3.2.

8.2.6.5 Domain Controller capable

CPCM devices are not required to be Domain Controller capable. However, implementers are recommended to include Domain Controller functionality in all devices that have sufficient resources to provide this function that are ADSE countable (DC controller capable devices are always CICF, see clause 8.2.7.2). Domain Controller capable devices are necessarily LM capable and thus have to implement the corresponding functions.

In addition, they have to:

- Implement protocols described in clause 8.6.3.3.
- Being able to generate an AD Secret and an ADID in the case where they create the AD. In particular, they have to maintain an AD creation index so that they do not allocate the same ADID twice to two created ADs with different AD secrets.
- Being able to initiate and securely maintain ADSE counters and ADSE ceiling.

- Being able to enforce all aspects of the ADSE method as described in clause 8.2.7.2, if this method is selected by the C&R regime.

Further, it is recommended that:

- Domain Controller capable devices have persistent power supplies (such as mains electricity), and adequate battery life for portable devices, since the DC function often needs to be available to respond to ADM messages.

- Domain Controller capable devices have persistent, secure storage to hold and protect valuable AD information.

EXAMPLE: An implementation might duplicate and sign the stored AD information to protect against single point of failure and allow recovery.

8.2.6.6 ADSE countable

For Countable Instance of CPCM Functionality (CICF) is decided by the C&R regime. If the standard method described in TS 102 825-7 [i.12] is used, CICF are instances that are Consumption Points or Export Points.

There is no extra capability to implement for a CICF. A CICF is necessarily AD aware but other capabilities are not mandatory.

EXAMPLE: A CICF need not be DC capable.

A CICF always counts for 1 in ADSE related protocols. A CPCM C&R regime may impose a device with high capabilities to implement several CPCM Instances, all being CICF so that this device counts for more than 1. How the splitting is done is up to the implementer. The different CICF in the device will behave independently and thus are not required to join a same AD at the same time.

8.2.7 Authorized Domain Size and Extent

Implementers are strongly recommended to review the specific requirements of the governing C&R Regime(s) prior to implementation of the ADSE method. Some C&R Regimes may use very different parameters and methods.

ADSE is intended to provide a rich enough set of evaluations so that the vast majority of users will not encounter a problem adding a new device to their AD. However, the parameters chosen by the C&R Regime will have a direct impact on the user experience, and implementations should ensure their UI design and consumer manuals are as clear as possible in this area.

Some C&R Regimes may include completely different ADSE techniques to meet specific commercial objectives.

8.2.7.1 ADSE counts and values

The standard ADSE method makes use of four ADSE counts and their associated ceilings which are maintained by Domain Controllers. These counts are:

- Total count, that counts the total number of CICFs that joined the AD, regardless of whether the Join was remote or local. The associated ceiling is total_ceiling. Total_count is incremented and decremented each time a Join or a Leave occurs. Once total_ceiling has been reached, no further Join is authorized (including if the quorum test passes) unless the ADMAAA authorizes it or a CICF leaves the AD.

NOTE 1: When the DC is split, total_count and total_ceiling are also split and thus do not correspond anymore to the total_number of CICF in the AD. This count can be obtained by adding total counts from each DC in the AD.

- Remote count that counts the number of CICFs that joined the AD remotely. The associated ceiling is remote_ceiling. Remote_count may be decremented if the CPCM Instance that joined remotely either becomes Local to the DC with which it Joined, (or the DC that maintains the list of CICFs that joined remotely in which it is included) and conditions for a Local Joining are met (see below) or Leaves through that same DC, whether the Leave is Local or Remote. When remote_ceiling has been reached, no further Remote Joins may be done unless the ADMAAA authorizes it or if the remote count is decremented.

NOTE 2: When the local_ceiling has been reached and the quorum test does not pass, a Local Join may be counted as Remote (if the remote_ceiling has not been reached), in which case the behaviour will be exactly the same as if the Join was actually Remote (i.e. the CPCM Instance identifier will be added to the list of CICF that joined Remotely).

- Local count that counts the number of CICFs that joined the AD locally. The associated ceiling is local_ceiling. local_count may be decremented when a Leave occurs and the Leaving CICF is not in the list of CICF that Joined Remotely. When local_ceiling has been reached, further Joining can happen if total_ceiling has not been reached: this is possible if the ADMAAA authorizes it, if the quorum test passes or if remote_ceiling has not been reached (in which case the Joining is viewed as Remote).

NOTE 3: If the total_ceiling is equal to the sum of local_ceiling and remote_ceiling, the quorum test can be skipped: when local_ceiling is reached, it is more efficient to directly count the joining on the remote_count; the consequence in terms of user experience will be the same.

- DC split count that counts the number of DC splits that occurred. The associated ceiling is DC split ceiling. DC split count can be decremented when the DC is remerged. Once the ceiling has been reached, no further split may occur unless the ADMAAA authorizes it or the DC split count is decremented.

- DC remote count, that counts the number of Remote DC transfers or Remote DC splits. The associated ceiling is DC remote ceiling. DC remote count cannot be decremented. Once the ceiling has been reached, no further split may occur unless the ADMAAA authorizes it.

In addition, each AD aware instance maintains the history count and the history ceiling that counts the total number of ADs of which the Instance was a member. This count is never decremented and once that ceiling has been reached, the CPCM Instance can no longer Join any other AD, unless the ADMAAA authorizes it. The history count is never decremented. For the sake of clarity, when a Blank Instance joins an AD with a DC, only the history count of the Blank Instance is verified and incremented.

All the ceilings are set at the time of device manufacture or upon initial software installation. Counts are initialized to zero (but local and total counts are immediately incremented by one when the AD is created).

Implementations have to protect the value of ADSE counters in terms of integrity. More particularly, measures have to be taken to prevent the replay of old counts. This can be achieved for instance by having in the DC a key stored in a secure read-write memory: the key is updated at each count change and used to sign all the counts. In an AD-aware instance that is not DC capable the same mechanism can be used or, in the case of hardware implementations, a one-time-programming bit can fused each time an AD join is done.

NOTE 4: A C&R regime may relax one of the above limits by setting the ceiling value to 0xFFFF, in which case implementations may decide not to support the associated constraints.

8.2.7.2 Standard ADSE method

TS 102 825-7 [i.12] proposes a standard ADSE method, which is optional. This means that a C&R regime may decide to adopt another ADSE method. However, this method has been designed to balance the interests and wishes of the different parties and has been agreed in principle by a number of industry participants.

The standard ADSE method builds upon several ADSE tools, namely:

- Domain History Membership (see clause 8.5.6) which is used as is, for all CPCM Instances, (including Instances that are not ADSE countable, see below).

- Quorum Test method (see clause 8.5.4) where the quorum percentage is computed against the current count of AD members that Joined Locally, the quorum percentage defining the floor of AD members that need to be present and Local to allow the Join once the local ceiling has been reached. In addition, the following recommended tools of the Quorum Test are used:

  - Allowing a Local Join when the local ceiling has been reached, the quorum test has failed, and there are still available Remote Join slots. The Join is then managed as a standard Remote Join.

  - Allowing DC to be Transferred, Split, Merged and Rebalanced with ceilings for the number of splits and the number of Remote Transfers. The last ceiling also includes Remote Splits. Local Transfers or Splits are not limited (but the total number of Splits is).

- Maintaining a list of Instances that Joined Remotely to possibly rebalance counts if such an Instance happens to become Local. This list may be Transferred, Split, Merged and Rebalanced.

- ADMAAA tool (see clause 8.5.8) is optionally used in addition to the two above tools.

The main benefit of the standard ADSE method is the notion of CICF (Countable Instance of CPCM Function) that are the only Instances which are counted (and hence concerned with the Quorum Test). Other Instances are not counted at all and hence any number of such Instances may Join the AD. The only limitation is that only Local Joins are authorized. The DC is not necessary for such a Join; it can be performed by the LM. The CPCM specification defines a CICF as an Instance being a Consumption Point and/or an Export Point. The rationale is that there is no reason to limit the number of Instances that are neither a CP nor an EP as they cannot be used to access the content. This also mitigates some of the existing issues with count based ADSE methods (see clause 8.5.2).

Domain Controllers have to be implemented in a CICF. It would be actually not very meaningful to control the counts from the AD from a non-countable Instance.

A C&R regime may decide that devices with high level Consumption and Export capabilities will count as more than one. In this case, the implementer will have to implement as many Instances as the number of counts required by the C&R regime, each Instance being a CP or an EP and counting as one. Then, the AD management will be totally transparent for the DC that will manage the different Join and Leave independently.

EXAMPLE: A pay-TV television receiver with multiple HDMI outputs designed to feed multiple large screens in a public place such as a sports bar may need to be treated as several CICFs under a given C&R regime. However, the C&R regime may decide that the number of CICFs counted is lower than the number of HDMI outputs.

8.2.7.3 Other ADSE methods

A CPCM C&R regime may choose to use another ADSE method. Such a method could be derived from other ADSE tools as described in TS 102 825-7 [i.12], or could be completely new. Implementation guidelines for such a method are out of the scope of the present document.

8.2.7.4 ADSE parameters for Multiple C&R regimes

Different CPCM C&R regimes may define different ADSE parameters. In this case, there are several implementation choices:

1) The different CPCM C&R regimes agree to a set of ADSE parameters dedicated for CPCM Instances complying with all of the regimes.

2) An implementer can build a single CPCM Instance acting as Domain Controller to govern a single AD that meets (and is certified for) the requirements of both (or all) C&R Regimes (i.e. the tightest of each condition). This is the best approach where the two C&R Regimes are broadly similar in their requirements, as it allows the greatest flexibility for movement of content.

3) An implementer can include multiple CPCM Instances within the device, each of which will act as a Domain Controller for a single AD, one for each C&R Regime. Each DC will obey the rules of a single C&R Regime. This allows each AD to make maximum use of the flexibility granted by the governing C&R Regime, however content will be constrained to those devices that include a CPCM Instance joined to the corresponding AD only.

4) Combinations of any of the above is also possible.

In general it would be better if C&R Regimes could agree a common set of ADSE parameters; however this is beyond the scope of the CPCM specification.

8.2.8 AD name

The AD name is a human readable string that is only used for user interface purposes. It is freely configurable by the user.

AD name may be chosen by default by the Domain Controller that created the AD or left to the user choice before creating the AD. Implementers should warn the user of potential conflicts in AD name and avoid duplication of the same AD name for multiple ADs. Default names should be chosen to be as reasonably unique and avoid obvious names that are likely to occur in a different household, such as "My Domain", or "Home", etc.

8.3 ADM State Machine

This clause only applies to ADM capable CPCM Instances.

8.3.1 Timeouts

The CPCM ADM specification does not provide normative values for timeouts in the State Machine as stated in clause 6.3.1.2. There are many possible types of networks and environments which will affect the latency of the system, especially when ADM protocols are run:

- Complex actions that affect multiple CPCM Instances, such as a Join which is being relayed by a Local Master to a remote Domain Controller, may also need more time to complete.
- Actions that may require a human response from the other device should allow enough time, and also advise the user that they may need to look at the other device display, (or equivalent), that may be located in a different room. Time should thus be given to the user to navigate between the different devices and undertake the tasks they have been given.
- Implementations may also wish to record the response time for other known CPCM devices, and modify their timeout values dynamically to match specific expected response times.

8.3.2 General Protocols

8.3.2.1 Discovery

The CPCM system is designed to be network-agnostic. While many implementations of CPCM will be running on a home network such as DVB-HN where UPnP is available, this will not always be the case. So the ADM specification provides a generic method that works for any network whether UPnP is available or not. In practice TS 102 825-9 [i.11] allows CPCM, including ADM, to make use of the UPnP infrastructure when it is present.

A CPCM Instance reconnecting to a network launches the Discovery Protocol by broadcasting a message Discovery_request and waits for responses within a given time limit.

8.3.2.2 Local Master and Domain Controller Messaging

Sometimes an AD Member or a DC will need to communicate with the LM. The discovery protocol is used to identify the current LM (if any), when its identity is not already known. If there is no LM present or if the LM whose identity is known does not respond, an LM Election is launched by the initiating CPCM Instance.

The same protocol is used when a specific Domain Controller is requested for an ADM action. The discovery protocol is used to verify whether this Domain Controller is actually connected. If it is not, the CPCM Instance may also try to go through the Local Master using the delegation mechanism (see clause 8.6.4). The Local Master may also need to discover the requested Domain Controller before proceeding.

NOTE: A Local Master may be capable of additional means of discovering the Domain Controller beyond the use of the Discovery protocol on a Local network. In fact, this is often necessary when the Domain Controller is Remote from the LM. Such additional means are out of scope of this multi-part deliverable. An example is described in TR 102 825-11 [i.8] (Hosted CPCM service scenario).

8.3.2.3 Connected AD member

A connected AD member may receive ADM protocol messages from time to time. There is no reason to leave a "sleeping" mode to respond to these messages, but if the device is sufficiently active, the Instance is normally expected to participate in the protocols as appropriate:

- It normally takes part in Discovery and LM Election protocols.
- It records changes that may be signalled after an AD update or the Election of a new LM (these records may be done in volatile storage).
- It starts Leaving, Transfer and Splitting Protocols upon invitation from its Device Application, or from another instance through an ADM invite message (after confirmation of its Device Application), assuming it meets the necessary conditions.
- Upon request from its device application, it sends a name change to a DC, possibly through a Local Master.

All other AD-related messages are ignored.

8.3.2.4 Connected Domain Controller

A connected Domain Controller, which is not a Local Master, may receive ADM protocol messages from time to time. The Domain Controller should normally react accordingly.

NOTE: Because of its importance to ADM, implementers need to consider how to maintain device responsiveness. The capabilities of a DC should not be adversely affected by going into a stand-by or a sleep mode, when possible. Obviously, a battery-powered device raises challenges. However, even a device on mains power needs to allow for rapid presentation of a suitable user interface when an ADM request arrives. Care should therefore be taken when designing the power management of a DC capable device.

Expected behaviour is as follows:

- It normally takes part in Discovery and LM Election protocols.
- It starts ADM protocols upon receiving the relevant Begin message.
- It records changes that may be signalled after an AD update or the Election of a new LM. AD update is performed in persistent storage while the LM identity update may be held in volatile storage. If it updated its own internal record, it re-broadcasts its latest records in an AD update indication message.
- It performs the AD name change upon request from another Instance after confirmation from its device application. It then informs all other Instances of the renaming through an AD update indication message.
- It proceeds with DC Rebalance or DC Merge protocols as a client upon request from its device application, or upon ADM invite message, after confirmation of its device application. A discovery protocol may be needed to find the relevant DC.
- It proceeds with the AD Leaving protocol upon invitation from its device application or from another device. It confirms first with its device application, and if it is not the sole Instance in the Domain, it is recommended to suggest to the user that they should first perform a DC merge or a DC transfer protocol, if possible.

All other AD-related messages are ignored.

8.3.2.5 Connected Local Master

A connected Local Master, which is not a Domain Controller, may receive ADM protocol messages from time to time. The Local Master has to react accordingly. Therefore, it may be appropriate to give up its LM responsibility before going into sleep mode, by launching a new LM election protocol. However, if the device power management allows sufficient responsiveness for LM functions, this may not be necessary.

Expected behaviour is as follows:

- It normally takes part in the Discovery protocol.
- It starts ADM protocols upon receiving a Join Begin or Leave Begin message.
- It records changes that may be signalled after an AD update. These records may be done in volatile memory. If it updated its own internal record, it re-broadcasts its latest records in an AD update indication message. It may also perform this update when receiving an AD update indication from a Remote DC and re-broadcast it locally.
- If it receives any message related to LM Election from the same AD, or if it receives another message originating from a different Local Master, it re-launches a new LM Election process. This is because such a situation indicates that there are several AD members acting as LM in the same Local Environment. This situation can sometimes occur in normal system operation, and is not really harmful, but needs to be resolved using the LM Election protocol.
- It acts as a proxy for AD change requests: it forwards the request to a DC and forwards any responses to the requesting Instance. If no response is received, it also informs the requesting Instance of the AD change failure. If the request comes from the device application, it also sends the request to a DC and informs the Device Application of the result.
- It performs the AD Leave protocol upon invitation from its device application or from another device. It confirms first with its device application, and then launches an LM Election to give up its LM role. It then proceeds normally with the AD Leave protocol, even if it is still the LM.
- If it is DC capable, it proceeds with DC Transfer or DC Split protocols as a client upon request from its device application or upon an ADM invite message after confirmation of its device application. A discovery protocol may be needed to find the relevant DC.

All other AD-related messages are ignored.

8.3.2.6 Connected Local Master and Domain Controller

The behaviour of a CPCM Instance that is both a Local Master and Domain Controller is identical to clause 8.3.2.4 except as follows:

- If it receives any message related to LM Election, it behaves as per clause 8.3.2.5.
- If the Device Application confirms a Leaving Request without proceeding to a DC Transfer or a DC Merge beforehand, it launches an LM Election Request with lowest priority before proceeding with the AD Leaving protocol.
- If some other specific DC is requested for some AD operation, it acts as a Local Master forwarding messages to the indicated DC.

8.3.2.7 Local Master Election

For non-CICF or non DC-capable CPCM Instances, the AD Join and AD Leave protocols rely on the presence of a Local Master that will allow the non-CICF Instance to Join or Leave or that will forward the requests to a DC for CICF Instances that are not DC capable. It is therefore very important to maintain the presence of a LM as much as possible. Since the LM role is ephemeral, it may happen that no current LM is available. If this absence is detected by an AD member, it will immediately launch a new LM Election. If it is detected by a non-AD member, the protocol described below may be used.

During a LM election, the AD member that launched the Election broadcasts its LM capability, or when specifically mentioned in this multi-part deliverable, the highest or the lowest level of the LM capabilities. Upon receiving this message, each LM capable Instance compares the received capability with its own capability. If it is the same or lower than the received one, it will not take part in the LM Election process. Else, its capability is higher and it broadcasts it to the other LM capable Instances. As a result, the LM capable Instance with the higher capability will be elected. This will be indicated to all present CPCM Instances by broadcasting the LM Election Indication message.

Implementations should endeavour to keep this behaviour transparent to the user, though it may be appropriate to include some form of trouble shooter mode to enable customer support to investigate problems.

An appropriate timeout to wait for an election to end on a normal home network would be 5 seconds from the arrival of the last election message.

LM capability is based on several factors, namely the CPCM versions (Instances with higher versions are more capable), AD capabilities and CPCM instance certificate identifiers, which are only used to avoid the election of several LMs when Instances with the same AD capabilities and CPCM versions are present. This can be changed dynamically, for example if the user, or a customer support staff, wishes to force a specific device to become the new LM. In this case, the AD capability factor is more important than the CPCM version, while the contrary applies in other cases.

As noted in the specification, the minimum and maximum values of LM Capability can only be used when it is necessary to force a device to stop or start being the LM respectively.

EXAMPLE 1: A device currently acting as LM is powering down. It triggers an LM election with itself showing LM Capability 0 to ensure that another device smoothly takes over the role.

EXAMPLE 2: A device has been configured with special information about the location of the remote Domain Controller. It triggers an election with itself showing LM Capability 4 so that it will become LM and be able to relay messages to the DC correctly.

Lowest capability is also used by non-LM capable Instances requesting an Election. If a LM capable Instance is connected, it will have a higher capability and will thus inform the non-LM capable Instance and a new LM will be selected. Else, the non-LM capable Instance will not receive any answer and know that no LM can be elected at that time.

To avoid absence of a LM on the network, it is recommended that a CPCM Instance that plays the LM role and that is about to be disconnected (because the user is about to take it away or to switch it off) launches a new LM Election process (taking part with the lowest priority level, see below,) in order that a new LM is elected, (if any LM capable Instance is connected). The LM Election process may only be launched by an AD member. Any LM election request message sent or broadcast by a non-AD member will be ignored. If a Blank Instance discovers that the AD it wants to Join has no LM, it will signal to one LM capable Instance that there is no LM using the error message (in response to the discovery response message) 'no LM or DC present: AD Join not possible'. This error message needs not to be sent if the Blank Instance has no intention of Joining the Domain.

The LM capable device will try to contact its current LM as described in clause 8.3.2.2. If the absence of LM is confirmed, the LM capable device will launch a new LM Election request and the Blank Instance will be informed of the new LM Election and will be able to proceed with the Join as shown in figure 20.

NOTE: The LM Election process will not be launched if the LM capable device detects that the LM is still connected. This process allows a faulty or malicious device to cause denial of service by permanently launching new LM election requests.

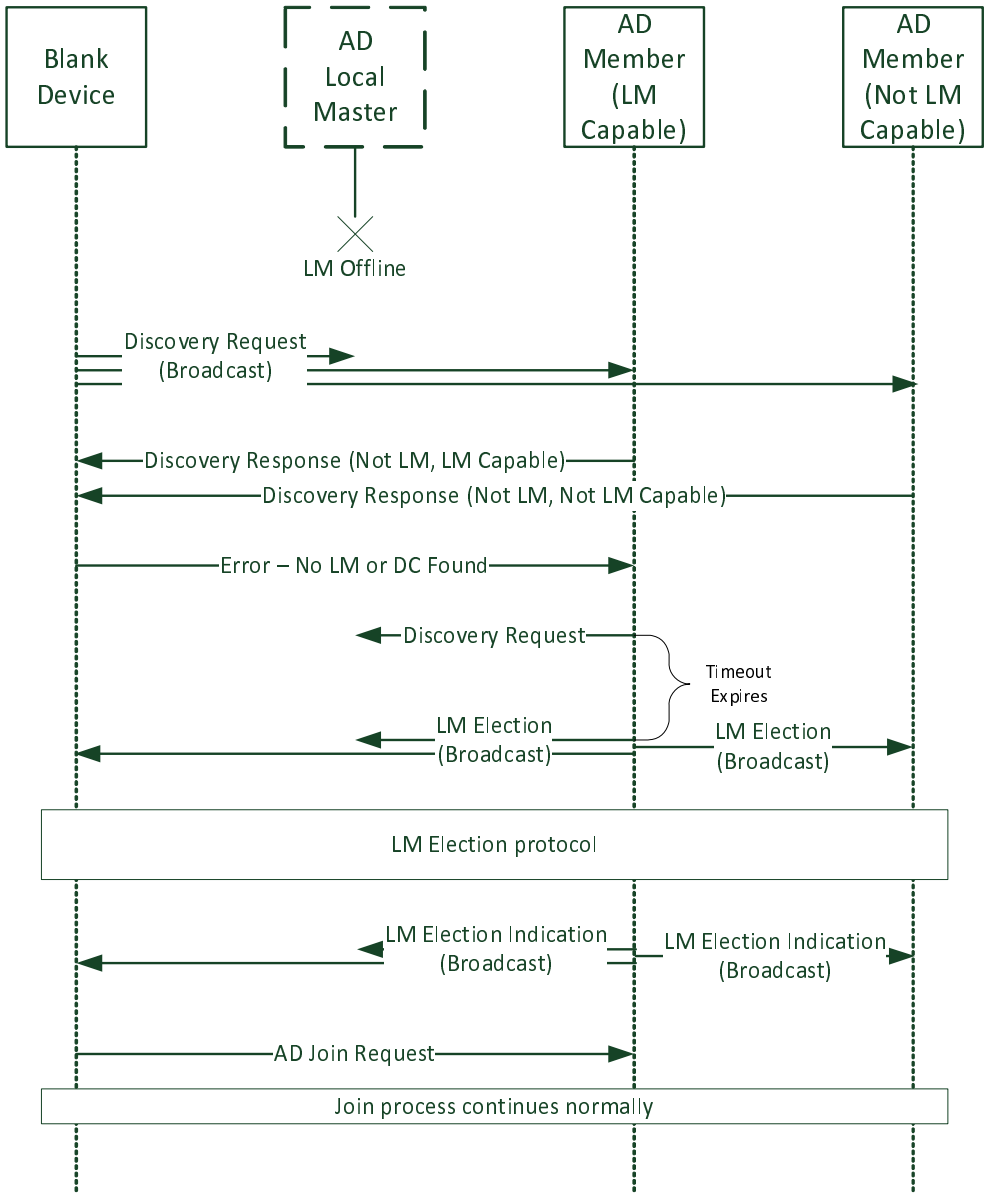


Figure 20: LM Election process launch by a non-AD member

8.3.2.8 AD Member Reconnection

The protocol applies to reconnecting AD members and Domain Controllers.

The goal of this protocol is to allow the reconnecting AD member or Domain Controller to resynchronize with the latest AD updates i.e. to get the latest AD internal record (see clause 8.3.4). It also allows the Domain Controller to become a Local Master if it has the greater capabilities.

At the end of the protocol, the reconnecting AD member will be in the following state:

- Connected AD member if a Local Master for its AD was present. In addition, it may have updated its AD internal record if the one from the LM was more recent.
- Connected Local Master or connected LM-DC (if the reconnecting CPCM Instance is a DC) if no Local Master was present. In this case, the AD internal record has not been updated.
- Connected DC or Connected LM-DC if a Local Master was present. In this case, the DC may have updated its AD internal record if the one from the LM was more recent, in which case it broadcasts an AD update Indication message once the update has been made. Final state is dependent upon a LM election process.

It may happen that the LM does not reply to AD update requests, for instance if it was disconnected in the meantime. In this case, the reconnecting CPCM Instance will launch a LM election process, as the first CPCM Instance to detect the lack of LM. It will present the lower priority in this process as it may not have the latest AD update. If it is however elected, this means that it is the only connected LM-capable AD member. If connected AD members have a more recent AD internal record, they may send it using an AD update response message to the connecting Instance so that the AD update occurs. Otherwise no AD update can occur. If it was not elected as LM, it re-starts the AD update process with the new elected LM.

At the end of the protocol, the LM may also have an updated AD internal record if the one received from the reconnecting CPCM Instance was more recent. In which case, it broadcasts the updated AD internal record.

As a consequence, at the end of the protocol, all connected AD members, LM or DC have an up to date and synchronized AD internal record.

Once reconnected, a CPCM Instance that had started an ADSE-related protocol that was interrupted (i.e. a CPCM Instance with recorded CIC identifier) will try to achieve the protocol if it detects the CPCM Instance with which the protocol was run. Further details are given in clause 6.3.3.4.

This protocol is one situation where there is a reasonably high chance of simultaneous transactions.

EXAMPLE 1: A home network segment that connects several devices becomes disconnected from the rest of the home network. When the segment is reconnected, all of the devices try to reconnect to the domain at the same time.

EXAMPLE 2: A home network recovers from a mains power interruption. Multiple devices start up and attempt to reconnect to the domain.

Implementers should consider using a multi-threaded approach to allow multiple reconnections in parallel.

8.3.3 ADSE Related Protocols

8.3.3.1 Atomic Transactions

An extensive analysis of this sequence was carried out, and the ADSE-related protocols defined in the specification are necessary to ensure that nothing is lost in the event that the protocol terminates unexpectedly. Firstly, it is important that every step is completed to ensure that secrets are only exchanged at the correct time, secondly it is important to ensure that unauthorised ADSE operations cannot be performed by interrupting communication during the process, and thirdly it is important to ensure that such an interruption does not adversely affect the AD, such as by reducing the number of future devices that can be added.

EXAMPLE: A user is having trouble joining a device to the AD. Perhaps the device is faulty, or there is an unstable communications environment. The user may make several attempts to join the device before giving up and calling for technical support. It would be a very bad experience to then find that the possible domain size had been irretrievably reduced by each of those unsuccessful attempts.

Thus ADSE related protocols, and the underlying atomic transactions, include every possible means to prevent any of these adverse consequences from occurring, even when one or both of the communicating devices is faulty or experiencing communications difficulties.

NOTE: The recovery mechanism works only if the two CPCM Instances involved in the ADSE related protocol are reconnected together. It is therefore recommended to retain the recovery information in non volatile memory as long as possible.

8.3.3.2 User Interface management

The CPCM specification does not specify the User Interface (UI); it is left to the implementer to deploy a design that is sufficiently user friendly. CPCM assumes integration of the ADM controls and signals with the device application in harmony with the Network Management System and its UI.

The design of the ADSE related protocols leaves the opportunity for the application to request confirmation to the user twice on the server side and twice on the client side. While it is not recommended to actually request confirmation from the user four times, it is recommended to request positive user action at least once on each side, in order to prevent unauthorised remote control of the device. Positive user action includes:

- Launching the protocol on the client device.
- Launching the protocol by an invitation from the DC. This is a positive action on the DC only.
- Accepting the protocol to proceed.
- AD unlocking.

It is strongly recommended that special operations like AD creation or AD deletion are specifically confirmed by the user, with very clear and understandable warnings describing the consequences of these actions. It would also be appropriate to protect such actions from being requested by children or other unauthorized persons.

NOTE: The reason for this extra care is because these actions can be very difficult or impossible to reverse should a mistake be made. For instance, if several ADs are created in a single household, this could result in acquired content being bound to different ADs and thus bad user experience. The deletion of one of the ADs to improve user experience may cause some AD-bound content to become unusable.

8.3.3.3 Blank Instance Connection

This protocol allows a connected Blank Instance to detect whether it may start an AD Join process. Circumstances under which this may happen are described in clause 8.2.3.

Upon request from its application or upon ADM invite message to proceed to Join, the CPCM Instance first checks with its Security Control as to whether it can proceed to Join (see clause 8.4.3). If not, it will ignore all other requests to Join an AD.

If the Join is permitted by the Security Control, it discovers any connected CPCM Instances. Several cases may then occur:

- No other CPCM Instance is connected. In this case, if it is DC capable, it proposes to its own Device Application and thus to the user as noted in clause 8.3.3.2 to create a new AD. If it is refused or of the Instance is not DC capable, it stops the protocol. If it is accepted, it proceeds to the AD creation (see clause 8.4.1) and broadcasts an AD update indication message.
- Other CPCM Instances are connected but they are all Blank. In this case, if it is not DC capable, it stops the protocol. If it is DC capable, it determines whether it has the lower CIC identifier amongst present DC capable CPCM Instances or not and if so requests to its Device Application to create a new AD and will then behave as described above. If it does not have the lower CIC identifier, it stops the protocol and should inform the user that another device will create the AD. Ideally the UI should tell the user which device to check.
  The CIC identifier mechanism is here to prevent two or more Blank Instances that would have been connected simultaneously to independently create new ADs, which would be confusing for the user and complicate content Acquisition.
- The CPCM Instance has a recorded ADID and an AD member from that AD was discovered. This means either that a Join already started with that AD but could not be completed, or that the Blank Instance was previously a member of that AD then left it without fully completing the Leave protocol.
  - If the original DC is present (i.e. the CPCM Instance with the recorded CPCM Instance Identifier), the Blank Instance tries to restart the protocol with the original DC.
  - If the original DC is not present, the Blank Instance sends the 'AD Join Begin' or 'AD Leave Begin' message to any LM or DC of the same AD that was discovered, requesting delegation with the original DC.
    - If no LM or DC is present but LM capable CPCM Instances are present, it may send an error message "no LM or DC present: AD Join not possible" in reply to the Discovery Response message to a LM capable Instance so that a new LM may be elected. It then waits for the LM Election.
    - If no LM capable CPCM Instance was discovered, it stops the protocol.

- If the AD was not discovered but other ADs were, the CPCM Instance may behave as described below but in case it proceeds to a Join, implementations are recommended to inform the user of the possible effects on the AD whose identifier was recorded (i.e. decreased device capacity).

- Protocol started upon ADM invitation and at least one AD member from the requested AD answered. In this case, once its device application has confirmed permission, it proceeds to the AD Join protocol with a LM or DC from that AD, if any is present. To that end, it sends an 'AD Join Begin' message. If no LM or DC are present but LM capable Instances are, it may try to re-launch the protocol by send error message "no LM or DC present: AD Join not possible" in reply to the Discovery Response message to a LM capable Instance. If no LM or DC from that AD is discovered, or if the application refuses, the protocol is stopped.

- CPCM Instance is self-managing and responses are received from at least one AD member. In this case, the CPCM Instance proposes to its device application to join the AD (if all responses were from the same AD) or to select between the available ADs. If the application confirms, it proceeds to the Join to the corresponding AD by sending an 'AD Join Begin' message to the applicable LM or DC. If several DCs are available, the Joining can be done with any of them but preference should be given to the LM as it normally has the highest capabilities. Alternatively, implementations may leave the choice to the user. If the application refuses, the protocol stops here. If no LM or DC is present for the selected AD, the user interface may signal the fact to the user and signal whether LM capable Instances are present. If not, the user interface may signal that a DC or a LM capable device should be connected to proceed to the Join with that AD. If one is present, the UI may propose the AD for Joining. If the AD is selected, the Blank Instance will first send the message "no LM or DC present: AD Join not possible" in reply to the Discovery Response message to a LM capable Instance such that a new LM is elected and the AD Join can be performed.

- Blank Instance is not self managing and responses are received from at least one AD member. In this case, the Blank Instance proposes to its device application to join an AD. If the application confirms, the CPCM Instance starts the AD Join protocol by sending (using CPCM broadcast) an 'AD Join Begin' message to one LM or DC in each AD (if several DCs are available for a given AD, same guideline as above applies). The broadcasting is necessary as the user cannot select the AD using the user interface or know whether several ADs are available. The selection will be made when the user will confirm the Join operation on the relevant LM or DC. If no LM or DC was present for the AD the user wanted, the user will not have performed any confirmation on any LM or DC and the protocol will timeout and stop. In this case, implementations may recommend the user to connect a DC or LM capable device. If the user (or the device application) refuses, the application may propose to the user to create a new AD if the Blank Instance is DC capable.

When a protocol has stopped, implementations may re-propose the AD Join upon reception of an LM election or an AD update Indication message. The protocol has probably stopped because no LM or DC was present and the message signals that a new one is available. For self-managing Instances, only messages for the selected AD can be taken into account in this case. Before asking confirmation to the application, CPCM Instances have first to check with their Security Control that they may Join an AD.

8.3.3.4 AD Joining

The AD Joining protocol allows a Blank Instance to become an AD member. It is run:

- Between the Blank Instance and the Domain Controller which is acting as the Local Master, if any.

- Between the Blank Instance and the Local Master if the Blank Instance is not ADSE countable.

- Between the Blank Instance and a Domain Controller if several Domain Controllers are discovered in addition to the Local Master (which is probably also a Domain Controller) and the Blank Instance selects that Domain Controller to run the protocol with.

- Between the Blank Instance and a (probably Remote) Domain Controller through a Local Master that handles the delegation messages in all other cases.

For a Blank Device, the simplest method is to always go through the Local Master that will determine which case applies. An advanced user interface may allow an expert user to choose between the above options (and mainly between the different Domain Controllers that were discovered). It is always preferable to join (if possible) with a Local Domain Controller since a Local Join will always be authorized provided that the maximum size of the AD has not already been reached.

To launch the protocol, the Blank Instance behaves as described in clause 8.3.3.2. If an AD Join is possible and is selected by the user, a message AD Join Begin has been sent to a Local Master or a Domain Controller. The Blank Instance waits for all replies to be received. Since the reply needs to be SAC secured, it may need to accept SAC establishment requests in the meantime. The following cases may occur:

- If it did not receive any AD Join Ready message, the protocol times out. This probably means the Local Master or the Domain Controller is no longer present, or that bad network connectivity prevents the normal protocol from running. The application should warn the user and the protocol stops here.

- The same behaviour is adopted if the only received message is a Transaction Rollback message. In this case, the Domain Controller refused the Joining.
  If one or more AD Join Ready messages are received together with the Transaction Rollback, the behaviour is the same as if no Transaction Rollback was received. This is because a Transaction Rollback may be sent by anyone, so a Denial of Service Attack would be easy to mount.

- The case where more than one AD Join Ready message is received may occur when the Blank Instance is not self-managing, in which case it might have sent several AD Join Begin messages. Thus the user has either confirmed on several devices, or some other entity confirmed (intentionally or by error) elsewhere. In this case, the protocol stops as there is no means to determine which AD is to be Joined. Implementations may choose to re-run the protocol a limited number of times (sending messages AD Join Begin to the same CPCM Instances) or to stop here. Before stopping the protocol, a Secured Transaction Rollback is sent to all the CPCM Instances that answered.

NOTE 1: There are exceptions where the protocol can proceed even though several AD Join Ready messages are received:

  - All the messages come from the same AD: in this case, there is no ambiguity on which AD will be Joined even for non self-managing CPCM Instances. The protocol can be continued (as described below) with any of the answering CPCM Instances. A Secured Transaction Rollback message is sent to the other responding CPCM Instances.

  - Messages come from different ADs but the Blank Instance is self-managing: in this case, only one AD Join Begin was sent and the protocol can be run (as described below) with the CPCM Instance to which the message was sent. A Secured Transaction Rollback message is sent to the other Instances.

- Else, the protocol proceeds with the CPCM Instance that responded. The CPCM Instance first verifies with its application that it can proceed to the Joining and then with its Security Control. If the Security Control accepts, the Blank Instance sends a message AD Join Commit. If it receives back a message AD Join Confirm, the AD key was successfully delivered but is not yet usable. The CPCM Instance asks its Security Control to enable the AD Secret and ends the protocol by sending a message AD Join Finish. The CPCM Instance is now an AD member. If no message AD Join Confirm is received, the CPCM Instance remains Blank without sending any (Secured) Transaction Rollback message. This is to enable the protocol to resume later.

The protocol is either run directly with a Domain Controller (which may also be a Local Master) or via a delegated Local Master.

In the former case, the Domain Controller behaves as follows:

- Upon reception of an AD Join Begin message:

  - If the requesting Instance is ADSE countable, it checks with its Security Control whether the Join is authorized or not. The ADSE method is run at that time (see clause 8.2.7.2).

  - If it is not ADSE countable, it just checks whether the Blank Instance is Local or not.

If either of the above tests succeeds, the DC also checks with its application whether the Joining is authorized. If the test fails or if the application refuses, a Transaction Rollback message is sent back, and if resources were reserved for ADSE, they are freed.

NOTE 2: If the AD is locked (see clause 8.2.4), application confirmation will always be refused.

If the AD Join Begin message was broadcast, user confirmation is mandatory because this implies that the request came from a non-self managed CPCM Instance. Broadcast and sent AD Join Begin messages have different message codes and are thus easily identified.

- If all tests are positive, the DC sends back an AD Join Ready message. If the DC receives back a Secured Transaction Rollback, it frees the reserved resources and stops the protocol. Else, if neither an AD Join Commit nor a Secure Transaction Rollback message is received, reserved resources are not freed so as to enable the protocol to resume later (see clause 6.3.3.4).

- If an AD Join Commit message is received, the AD Secret can be delivered by the Security Control. After the delivery, an AD Join Confirm message is sent. If the Joining Instance was not ADSE countable, the protocol stops here (even though an AD Join Finish message may be received). Else, it has to wait for the AD Join Finish message in order to free the resources reserved by its Security Control and to broadcast a message AD update Indication that informs every present AD member of the Joining.

NOTE 3: The Domain Controller also sends the AD Update Indication message to all other Local Masters it is in communication with.

If run between the Blank Instance and the Local Master, the protocol can be described as follows:

- If the Blank Instance is not ADSE countable, the protocol is run without any involvement of a Domain Controller. The Local Master establishes a SAC with the requesting Instance, checks it is Local and requests confirmation from the LM's application.

  - If anything fails, a message Transaction Rollback (if the SAC has not yet been established) or Secured Transaction Rollback (if it is established) is sent.

  - If all tests pass, a message AD Join Ready is sent. If no response is received to this or if a message Secured Transaction Rollback is received, the protocol stops here. Otherwise, a message AD Join Commit is received. The Local Master asks its Security Control to send the AD Secret and sends back then a message AD Join Confirm. The protocol then stops, whether an AD Join Finish message is received or not.

- If the Blank Instance is ADSE countable, the protocol will be delegated to a Domain Controller. First, the Local Master will try to establish a SAC with the Blank Instance. If this fails, a transaction rollback message is sent. Else, a delegated AD Join Begin message is sent to a Domain Controller. The Domain Controller will behave as described above (but will establish the SAC with the Local Master rather than with the Blank Instance).
  If the Local Master receives back a Transaction Rollback message, or if no response is received at all from the DC, a message Transaction Rollback is issued to the Blank Instance. Else, it delegates the AD Join Ready message to the Blank Instance.
  If it receives back a Secured Rollback Transaction message from the Blank Instance, it delegates it to the Domain Controller. If no response is received at all, it stops the protocol without rolling back the transaction, because the requesting Instance might have committed resource
  Otherwise, it receives an AD Join Commit message from the Blank Instance that it delegates to the Domain Controller. The Domain Controller confirms to its Security Control that the AD Join is about to occur so that ADSE counts may be updated. It then sends back an AD Join Confirm message to the Local Master. At that time, the Local Master asks its Security Control to deliver the AD Secret and then generates an AD Join Confirm message to the Blank Instance. If no response was received from the DC or if the Blank Instance does not reply, the protocol stops here.
  When the Local Master receives a message AD Join Finish from the new AD member, it delegates the message back to the DC. The DC will broadcast an AD Update Indication carrying the new AD internal record that will be re-broadcast locally by the Local Master.

In any case, time elapsed between the sending of the AD Join Begin message by the Blank Instance and the receiving of the message AD Join Ready may be quite long, as the Local Master or the Domain Controller will always perform at least a proximity test and a SAC establishment before replying. Furthermore, it may perform additional ADSE checks such as the quorum test or the ADMAAA protocol. Also, it may be necessary for the user to visit another room, or give instructions to another family member to confirm the Joining both on the Blank Instance and the Domain Controller.

8.3.3.5 Instance Leaving the AD

The AD Leaving protocol allows an AD member to become a Blank Instance. This may be useful when the Joining was made by mistake or when the user wants to decommission, sell or give away the device, or uninstall the CPCM Instance from a device such as a PC. For ADSE countable Instances, it also ensures that the ADSE counts are updated correctly to recover the previously committed AD membership resources.

The protocol is started by user interaction with the device application. This may be initiated either on the Leaving Instance device itself or on another device using an ADM Invite (Leave) message.

The Leaving Instance first checks whether a LM or a DC is present. If not, it launches a LM election where it is sets its LM priority to the lowest value, in order to avoid being elected (since it is about to Leave the AD). If it is itself the LM, it should also re-launch a LM election in order to give up this role, though this is not essential. It then checks with its application (and maybe also with the user) whether it can proceed with Leaving the AD. This confirmation avoids a faulty or malicious device from causing all AD members to leave. If several DCs were discovered, the implementation may allow the user to select which DC the protocol is run with. Alternatively, this DC selection might be made automatically by the implementation itself, in which case a good choice is a Local DC which has low ADSE counts. If the application refuses, the protocol stops here, and the CPCM Instance remains an AD member.

The Leaving Instance then sends an AD Leave Begin message to the selected DC. If no answer is received or a Transaction Rollback message is received, the protocol stops here. Else, an AD Leave Ready message is received.

The Leaving Instance requests confirmation from its application and its security control to disable its AD Secret. If both succeed, it sends an AD Leave Commit message. Its status is still AD Member at that step. Else, it sends a Secured Transaction Rollback message and the protocol stops.

Finally, it waits for the AD Leave Confirm message from the DC. Upon receiving it, the Leaving Instance informs its Security Control that the AD Leaving was successful so that the Security Control erases the AD secret, and sends back an AD Leave Finish message. It is now a Blank Instance. If no AD Leave Confirm message is received, it behaves as a Blank Instance but will have to resume the protocol as soon as possible in order to ensure that the ADSE counts have been correctly updated (see clause 6.3.3.4).

The protocol is either run directly with a Domain Controller (which may also be a Local Master) or via a Local Master using delegation.

In the former case, the Domain Controller behaves as follows:

- Upon reception of an AD Join Leave message, if the requesting Instance is ADSE countable, it notifies its Security Control that an AD member is about to Leave the AD. The DC then checks with its application whether the Leaving is authorized and establishes a SAC with the Leaving Instance. If the application refuses or if the SAC establishment fails, a Transaction Rollback message is sent to the Leaving Instance and Security Control is notified that the Leave is abandoned and the protocol stops here.

- If the SAC is established and the application confirms the Leave, the DC sends back an AD Leave Ready message. If the DC receives back a Secured Transaction Rollback, it notifies the Security Control that the Leave is abandoned and stops the protocol. Else, if no AD Leave Commit message is received, Security Control is not notified in order to enable a protocol resume (see clause 6.3.3.4).

- If an AD Leave Commit message is received, the DC informs its Security Control that the Leave is to be performed and indicates whether the Leaving Instance is Remote or not.

NOTE: If the DC is aware that the Leaving Instance was originally Joined Remotely, and is now Local, and that this change has not yet been reflected in ADSE counts, it may be appropriate to indicate that the device is still Remote to the security control so as to recover the more limited Remote Join count.

- Then, an AD Leave Confirm message is sent. It waits for the AD Leave Finish message in order to broadcast an AD update Indication message to inform every present AD member of the Leaving. This uses CPCM broadcast to reach every LM that the DC is in communication with. It also informs its security control of the success of the Leaving.

If run via a Local Master, the protocol can be described as follows:

- If the Leaving Instance is not ADSE countable, the protocol is run without any involvement of a Domain Controller. The Local Master establishes a SAC with the requesting Instance, and requests confirmation from the LM's device application. If either is refused, a Transaction Rollback message (if the SAC has not yet been established) or Secured Transaction Rollback (if it is established) message is sent to the Leaving Instance and the protocol stops. Else, an AD Leave Ready message is then sent. If no response is received or if a Secured Transaction rollback message is received, the protocol stops here. Else, an AD Leave Commit message is received. The Local Master asks its Security Control to delete the AD Secret in the Leaving Instance and then sends an AD Leave Confirm message. The protocol stops here, whether an AD Leave Finish message is received or not.

- If the Leaving Instance is ADSE countable, the protocol will be delegated to a Domain Controller. First, the Local Master will try to establish a SAC with the requesting Instance. If this fails, a transaction rollback message is sent. Else, a delegated AD Leave Begin message is sent to the chosen Domain Controller. The Domain Controller will behave as described above, with the exception that it establishes the SAC with the Local Master (and not the Leaving Instance).
  If the Local Master receives back a Transaction Rollback message or if no response is received at all, a Transaction Rollback message is sent to the Leaving CPCM Instance. Else, it delegates the AD Leave Ready message to the Leaving Instance. If the LM receives back a Secured Rollback Transaction message, it delegates it to the Domain Controller. If no response is received at all, it stops the protocol without rolling back the transaction as the Leaving Instance might have already disabled its AD Secret.
  Else, it receives an AD Leave Commit message that is delegated to the Domain Controller. The Domain Controller confirms to its Security Control that the AD Leave occurs so that ADSE counters may be updated. It sends then back an AD Leave Confirm message to the Local Master. At that time, the Local Master ask its Security Control to proceed to the AD Secret deletion protocol and sends an AD Leave Confirm message to the Leaving Instance. If no response was received from the DC or if the Leaving Instance does not reply, the protocol stops here.
  When the Local Master receives a message AD Leave Finish from the newly Blank Instance, it delegates the message back to the DC. The DC will broadcast an AD Update Indication message carrying the new AD internal record that will be re-broadcast locally by the Local Master. It also informs its security control of the success of the Leaving.

8.3.3.6 Domain Controller Transfer

The Domain Controller Transfer Protocol allows a user to choose which DC capable device will become the Domain Controller. This may be useful when the user wants to re-purpose or dispose of the device that is the current Domain Controller or when the user has purchased a more capable device than the current Domain Controller.

The Domain Controller Transfer protocol does not make use of the delegation mechanism. As the CPCM Instance is about to become a Domain Controller, it needs to be Domain Controller capable and have the capability to discover the current Domain Controller(s) and to communicate directly with it. However, before starting the protocol, the requesting AD member will check whether there is a current Local Master, and, if not, launch a new LM Election process taking part with higher priority so as to be the Local Master after the Election process.

Domain Controller transfer is initiated by user interaction on the device application or by invitation from another CPCM Instance.

Before starting the process, the AD member requests its device application to confirm. If not confirmed, the process is immediately stopped. At this point, the application may also offer the user a choice of the DC to be transferred, if several are available. To that end, the CPCM Instance may run a CPCM Discovery protocol. If the transfer is confirmed, the AD member sends a DC Transfer Begin message to the DC to be transferred.

Upon receiving the DC Transfer Begin message, the DC first checks that the requesting CPCM Instance is DC capable. It also requests its device application and its security control to confirm the transfer. If any of these verifications fail, the transfer is stopped and a transaction rollback message is issued. Else, a DC transfer ready message is sent to the requesting instance.

Upon receiving the DC Transfer Ready message, the requesting CPCM Instance checks with its device application whether it can proceed and asks its security control to prepare to the DC transfer. If both the device application and the security control confirm, it sends a DC transfer commit message. Else, it sends a secured transaction rollback message and stops the protocol. If the DC Transfer Ready message is not received or if a transaction rollback message is received, the protocol also stops.

Upon receiving the DC Transfer Commit message, the Domain Controller asks its Security Control to start the transfer. From this time, this Instance stops being a DC: even if the protocol does not terminate, it will then behave as a simple AD member (or, if applicable, a Local Master). If the security Control confirms, it sends a Domain Controller Transfer Confirm message. Else, it stops the protocol without sending any rollback message and will not cancel any resources as the protocol may restart later. If the Domain Controller does not receive the Domain Controller Transfer Commit message, the protocol also stops without rolling back the transaction. If a secured transaction rollback message is received, the Domain Controller informs its security control to free the committed resources and stops the protocol.

Upon receiving the DC Transfer Confirm message, the requesting CPCM Instance informs the Security Control and becomes the actual Domain Controller. It informs the former Domain Controller by using a Domain Controller Transfer Finish message. It also broadcasts an AD Update Indication message to inform each AD member of the DC transfer and proceeds to a new LM Master Election. If the DC Transfer Confirm message is not received, the protocol stops here and the CPCM Instance has not become a Domain Controller. It may look in the received list of CICFs that Joined remotely to determine whether its own identifier is present, and, in such a situation, will attempt to remove itself from this list (as it is Local to itself). This will be possible if the local count has not reached local ceiling or if the quorum test passes.

Upon receiving the DC Transfer Finish message, the former Domain Controller informs its Security Control that the transfer was successful and that allocated resources can thus be freed, it stops the protocol here. It also stops the protocol here if no DC Transfer Finish message is received (but resources will not be freed to allow for resuming the protocol).

8.3.3.7 Domain Controller Splitting

The Domain Controller Splitting Protocol allows a user to split the Domain Controller function over two DC capable CPCM Instances (the former Domain Controller and another one). This may be useful for users who have an AD which spans two different LANs, typically a main house and a second house. This way, the local parts of the AD will be able to grow separately without taking on the Remote Joining counts.

The Domain Controller Splitting protocol does not make use of the delegation mechanism. As the CPCM Instance is about to become a Domain Controller, it is Domain Controller capable and has the capabilities to discover the current Domain Controller and to communicate directly with it. However, before starting the protocol, the requesting AD member will check whether there is a current Local Master, and, if not, will launch a new LM Election process taking the highest priority so as to be the Local Master after the Election process.

Before starting the process, the AD member requests its device application to confirm. If not confirmed, the process is immediately stopped. At that step, the application may also offer the user the choice of the DC that will be split, if several are available. To that end, the CPCM Instance may run a CPCM Discovery protocol. If the splitting is confirmed, the AD member sends a DC Split Begin message to the DC to be split.

Upon receiving the DC Split Begin message, the DC first checks that the requesting CPCM Instance is DC capable. It also requests its device application and its security control to confirm the split. If any of this verification fails, the split is stopped and a transaction rollback message is issued. Else, a DC split ready message is sent to the requesting instance.

Upon receiving the DC Split Ready message, the requesting CPCM Instance checks with its device application whether it can proceed and how the split is done (i.e. how the different ADSE counts and ceilings are split). The splitting may be decided by some default software process (e.g. one third of all counts and ceilings are given to the future new DC) or left for the user to decide (a C&R regime may impose restrictions on what the user may decide, see TR 102 825-13 [i.5]). It may also offer the Device Application the ability to transfer part or the entire list of CICFs that Joined Remotely during the Split. It also asks its security control to prepare for the DC Split. If both the device application and the security control confirm, it sends a DC Split Commit message. Else, it sends a secured transaction rollback message and stops the protocol. If the DC Split Ready message is not received or if a transaction rollback message is received, the protocol also stops.

Upon receiving the DC Split Commit message, the Domain Controller asks its Security Control to proceed to the splitting. If the security Control confirms, it sends a message Domain Controller Split Confirm. Else, it stops the protocol without sending any rollback message and does not cancel any resources as the protocol may restart later. If the Domain Controller does not receive the Domain Controller Split Commit message, the protocol also stops without rolling back the transaction. If a secured transaction rollback message is received, the Domain Controller informs its security control to free the committed resources and stops the protocol.

Upon receiving the DC Split Confirm message, the requesting CPCM Instance informs the Security Control and becomes a Domain Controller. It informs the split Domain Controller by using a Domain Controller Split Finish message. It also broadcasts an AD Update Indication message to inform each AD member of the DC Split. If the DC Split Confirm message is not received, the protocol stops here and the CPCM Instance is not yet a Domain Controller.

Upon receiving the DC split Finish message, the other Domain Controller informs its Security Control that the splitting was successful and that allocated resources can thus be freed, it broadcasts an AD Update Indication message to inform its local cluster about the splitting and stops the protocol here. It also stops the protocol here if no DC Split Finish message is received (but resources will not be freed to allow resuming the protocol and no AD update will be signalled).

8.3.3.8 Domain Controller Merging

The Domain Controller Merging Protocol allows a user to cancel the splitting of one Domain Controller. Other usages may also be found such as the rebalancing of DC counts between two Remote, unconnected locations: a Domain Controller is split in a given location to a DC capable device with the values to be re-balanced. The new DC is physically transported to the other location and re-merged to the DC of that location.

The Domain Controller Merging protocol does not make use of the delegation mechanism. As the initiating CPCM Instance is a Domain Controller, it has the capabilities to discover the current Domain Controller and to communicate directly with it. However, before starting the protocol, the requesting DC Controller will check whether there is a current Local Master, and, if not, launch a new LM Election process taking the highest priority so as to be the Local Master after the Election process.

Before starting the process, the merging DC (i.e. the DC that will be merged to another) requests its device application to confirm. If not confirmed, the process is immediately stopped. At this point, the application may also offer the user a choice of the DC with which the merging will be done, if several are available. To that end, the CPCM Instance may run a CPCM Discovery protocol. If the merging is confirmed, the merging DC controller sends a DC Merge Begin message to the merged DC (i.e. the DC into which the other DC is merged).

Upon receiving the DC Merge Begin message, the merged DC first checks that the requesting CPCM Instance is actually a DC. It also requests its device application to confirm the merge. If this verification fails, the merge is stopped and a transaction rollback message is issued. Else, a DC merge ready message is sent to the requesting Instance.

Upon receiving this message, the requesting DC checks with its device application whether it can proceed to the merging. It also asks its Security Control to prepare for the DC merge. If both the device application and the security control confirm, it ceases to be a DC and sends a DC merge commit message. Else, it sends a secured transaction rollback message and stops the protocol. If the message DC Merge Ready is never received or if a transaction rollback message is received, the protocol also stops.

Upon receiving the DC Merge Commit message, the Domain Controller asks its Security Control to proceed to the merging. If the Security Control confirms, it sends a Domain Controller Merge Confirm. Else, it stops the protocol without sending any rollback message and still retaining resources as the protocol may restart later. If the Domain Controller never receives the message Domain Controller Merge Commit, the protocol also stops without rolling back the transaction. If a secured transaction rollback message is received, the Domain Controller informs its security control to free the committed resources and stops the protocol.

Upon receiving the DC Merge Confirm message, the requesting CPCM Instance informs the Security Control of the successful Merge. If it was a Local Master, it launches a new LM Election process so that the most adapted CPCM Instance becomes the new LM (it may still be elected). Else, it becomes a simple connected AD member. It informs the Domain Controller by using a Domain Controller Merge Finish message. If the DC Merge Confirm message is not received, the protocol stops here and the CPCM Instance is no longer a Domain Controller. A LM Election is launched if it used to be the Local Master.

Upon receiving the DC Merge Finish message, the merged Domain Controller informs its Security Control that the merging was successful and that allocated resources can thus be freed, it broadcasts an AD Update indication message to inform its local cluster about the merging and stops the protocol here. It also stops the protocol here if no DC Merge Finish message is received (but resources will not be freed to allow for the resuming of the protocol and no AD update will be signalled).

8.3.3.9 Domain Controller Rebalancing

The Domain Controller Rebalancing Protocol allows a user to transfer some ADSE counts or some identifiers present in the list of CICFs that Joined Remotely from one DC to another. This might be useful if, for instance, the DC at the main location has no more join counts left while the one at the second house has remaining possible joins, and the user is more likely to add new devices to the main location.

The Domain Controller Rebalancing protocol does not make use of the delegation mechanism. As the initiating CPCM Instance is a Domain Controller, it has the capability to discover another Domain Controller and to communicate directly with it. However, before starting the protocol, the requesting DC Controller will check whether there is a current Local Master, and, if not, launch a new LM Election process taking the highest priority so as to be the Local Master after the Election process.

Before starting the process, the Rebalancing DC (i.e. the DC initiating the protocol) requests its device application to confirm. If not confirmed, the process is immediately stopped. At that step, the application may also offer the user a choice of DC with which the Rebalancing will be done, if several are available. To that end, the CPCM Instance may run a CPCM Discovery protocol. If the Rebalancing is confirmed, the AD member sends a DC Rebalance Begin message to the Rebalanced DC (i.e. the other DC).

Upon receiving the DC Rebalance Begin message, the Rebalanced DC first checks that the requesting CPCM Instance is actually a DC. It also requests its device application to confirm the Rebalancing. If this verification fails, the rebalancing is stopped and a transaction rollback message is issued. Else, a DC Rebalance Ready message is sent to the requesting instance.

Upon receiving this message, the requesting DC checks with its device application whether it can proceed to the Rebalancing. It also asks its security control to prepare for the DC Rebalancing. The rebalancing Domain Controller chooses at that step the amount of rebalancing to be done for both ADSE values and the list of CICFs that joined remotely. This can be performed by heuristics (e.g. trying to have similar number of remaining slots in both DCs) or upon interaction with the device application. If both the device application and the security control confirm, it sends a DC Rebalance Commit message. Else, it sends a secured transaction rollback message and stops the protocol. If the message DC Rebalance Ready is not received or if a transaction rollback message is received, the protocol also stops.

Upon receiving DC Rebalance Commit message, the rebalanced Domain Controller asks its Security Control to proceed to the Rebalancing. If the security Control confirms, it sends a Domain Controller Rebalance Confirm message. Else, it stops the protocol without sending any rollback message and still retaining resources as the protocol may restart later. If the Domain Controller does not receive the Domain Controller Rebalance Commit message, the protocol also stops without rolling back the transaction. If a secured transaction rollback message is received, the Domain Controller informs its security control to free the committed resources and stops the protocol.

Upon receiving the DC Rebalance Confirm message, the requesting CPCM Instance informs the Security Control of the successful rebalance. Then, it informs the rebalanced Domain Controller by using a Domain Controller Rebalance Finish message and broadcasts its new ADSE values in an AD Update Indication message. If the DC Rebalance Confirm message is not received, the protocol stops here.

Upon receiving the DC Rebalance Finish message, the rebalanced Domain Controller informs its Security Control that the Rebalancing was successful and that allocated resources can thus be freed, it broadcasts its new ADSE values in an AD Update indication message to inform its local cluster about the Rebalancing and stops the protocol here. It also stops the protocol here if no DC Rebalance Finish message is received (but resources will not be freed to allow for the resuming of the protocol and no AD update will be signalled).

8.3.4 AD Internal Record

The AD internal record is a shared data structure used by Domain Controllers to record meaningful AD information. It is transmitted to all other AD members that may choose to store it. Local Masters and Domain Controllers record it, and it is recommended for LM capable Instances to also record it. Having copies of the AD internal record in many places increases the chances of retaining the most recent one in the event of DC failure. This information may then be used by a C&R regime to regenerate a DC with the same capabilities and hence avoids the user losing some AD capabilities (or having to create a new AD with no access to much of the content that was recorded previously).

An AD internal record is maintained by each DC of the AD. Each internal record is identified through an index and each CPCM Instance recording AD internal records only keeps the one with higher index for each DC. Each time an Instance connects, if it stores AD internal records, it sends its current list to the LM. The LM compares the received records with its own records. If some of the received records are more recent, the LM will update the corresponding records and broadcast the list to other AD members. If some of the received records are less recent, it sends back a message carrying the list of corresponding newer records so that the reconnecting Instance may update them.

Each AD internal record is identified through the use of a DC local id which is relative to the AD. It is actually not possible to use the DC CIC identifier as there would be no means to make the link between the internal records of a DC before and after it is Transferred to another CPCM Instance. Thus, before proceeding to an update, a LM or an AD member first looks at the DC local id and then at the index.

DC local id is not changed for a DC Transfer. An instance that has become DC after a split is allocated a new DC local id by the splitting DC.

DC local ids cannot normally be reused within an AD. DC local id may be allocated randomly, or the C&R regime may define some policy. The approach taken needs to minimize the risk of collisions.

EXAMPLE: The splitting DC may allocate to the Split DC a range of local ids that will be for its exclusive use.

A list of currently used DC id and the one that have been merged is included in each AD internal record. The ones that have been merged cannot be re-used until all DCs have been informed of the merge, which is not easy to determine; else there would be no means to distinguish between the AD internal record of the merged DC and the one of the new DC (which will probably have first a lower index and will not then be retained). Local Masters and AD members may delete the AD internal records relative to Merged DCs. A DC local id present in the list of current DCs or in the list of the merged DC cannot be reused.

Each time the DC proceeds to a Join, Leave, DC Transfer, Split, Merge or Rebalance operation or that AD is renamed; it updates its AD internal record accordingly. It also updates its list of current and merged DC local ids each time it receives an updated AD internal record that contains an updated DC local id list. Each time it has updated its internal record or it has received one or more updated internal record, it broadcasts the list of updated internal records. It also deletes internal records relative to merged DCs.

When a DC becomes the unique DC of the AD, the risk of lack of synchronization vanishes: it empties the merged DC local id list and erases corresponding internal records. Local Masters receiving an updated internal record where there is unique current DC local id and the list of merged DC is empty may also delete all the previous internal records. This is the only exception where DC local ids may be reused.

8.4 Security Control Behaviour for ADM

The Security Control is responsible for the security state of the whole CPCM instance. Hence, attention has to be taken to ensure that it is not in an unknown state. If several protocols are run in parallel by the same DC, the security contexts of each protocol are to be handled separately and in a non ambiguous way.

8.4.1 AD Creation

When the Security Control is requested to proceed with the creation of a new AD, it first checks whether the history_count has reached its ceiling, in which case the creation is not allowed.

AD Secret is picked at random and stored securely. All ADSE counts (except the history_count) are initialized to zero and local_count and total_count are incremented by 1 (since the AD already has one member which is DC capable). All ceilings (except the history_ceiling) are also initialized according to the values set by the applicable C&R Regime. history_ceiling is incremented by 1.

An ADID is also created based on the CIC identifier followed by a byte index. This avoids creating different ADs with the same ADID, which might cause bad user experience. This implies that DC capable devices have to be able to securely maintain the value of such an index. If the specification allows for the index cycling, this can happen only if the history_ceiling value set by the C&R Regime is higher than 255.

NOTE: The AD name is handled outside of the Security Control as it is only used for UI purposes. It might, however, be initialized at the point of AD creation by the application either to a default value (which can be based on the current index) or to the user choice.

8.4.2 DC Leaving the AD

A user may request a DC to leave the AD. In this case, it is recommended to first determine whether other DCs or other DC capable Instances are connected in order to propose to the application a Merge or a Transfer of the DC before the Leaving occurs. Else, the device application is recommended to explain the possible consequences of the DC Leaving to the user, namely partial loss of AD capabilities (if there are other DCs – this can be determined by comparing current ADSE ceilings with the default ones defined by the C&R regime) or total loss (if it was the only DC) and maybe also loss of AD-restricted content (if there are no other AD members, this can be determined by checking whether total_count is 1 or not).

If the user confirms anyhow, the AD Secret and the ADID are erased and ADSE values are set to 0. It also ceases to be a Domain Controller and becomes a Blank Instance.

8.4.3 AD Joining

8.4.3.1 Blank Instance

Before starting the AD Joining protocol, a Blank Instance first checks with its security control whether the Join is permitted. At this point, the security control checks whether the Instance is ADM capable and whether history_ceiling has been reached.

Then, the security control is involved in the SAC establishment initiated by the LM (if the protocol is delegated) or a DC. This is always the case, even if the SAC already exists. At the same time, it will probably have to respond to proximity tests as the DC will have to determine whether the Joining is Local or not.

Before sending the AD Join Commit message, the Security Control checks that the context meets all the conditions (i.e. SAC established with a LM or DC, DC is a DC capable device that is a CICF). If all tests succeed and if the Joining Instance is a CICF, then the CIC identifier of the DC controller and the corresponding ADID with which the protocol is run is recorded in order to enable the protocol to resume if an error were to occur. It will then be ready to receive and store the AD secret. At this point, the AD secret is not yet usable.

Finally, upon confirmation of the AD management (i.e. when AD Join Confirm is received), it will enable the AD secret and become an AD member. It also erases the stored CIC identifier, if any.

8.4.3.2 Domain Controller

There are two different circumstances under which a DC is involved in a Joining protocol:

- The protocol is run with a non ADSE countable Instance. In this case, the role of the DC is mostly the same as the one of a simple Local Master: Upon reception of the AD Join Begin message, it establishes a SAC with the Blank Instance and checks whether the Instance is ADM capable. Then, upon reception of the AD Join commit message, the Security Control delivers the AD Secret using the established SAC.

- The protocol is run with an ADSE countable Instance, in which case the role of the DC is fundamental as it has to enforce the ADSE method.

The description below applies for an ADSE countable Instance where the C&R regime applies the standard ADSE method described in TS 102 825-7 [i.12].

When the DC receives an AD Join begin from a CICF, it first checks with its security control whether the Join is allowed. The security control enforces the ADSE method (see clause 8.2.7.2 when the standard ADSE method is used). It thus determines whether the Instance is Local or Remote and determines the relevant behaviour. If the message was delegated and the DC cannot discover the requesting Instance, the Join is treated as Remote. The DC may perform a quorum test at that time.

If the Join is authorized, the DC records the CIC identifier of the requesting Instance and whether the Join will be Remote or Local. The CIC identifier is kept to allow the DC to behave correctly should another Join Request be received before this protocol is completed or to allow the protocol to be resumed.

The DC then establishes a SAC with either the requesting Instance, in which case it checks that the AD capability of the certificate matches the one of the message, or with the LM if the message was delegated. If the DC knows how to address the Blank Instance, it may decide to establish the SAC directly with this Instance and stop using the delegation process.

Then, when the DC receives the AD Join commit message, the security control updates the ADSE count and delivers the AD secret or asks the LM to proceed to the delivery in delegation mode. It is essential to increment the counts before sending the message, as the reverse operation could lead to an attack where the AD Secret is delivered without modifying the counts. If the Join is remote, the list of CICF that Joined Remotely is updated accordingly.

Finally, upon receiving the AD Join Finish message, the DC knows the protocol was successful and requests the security control to erase the kept CPCM instance certificate identifier.

In addition, each time a DC encounters an Instance in the list of CICF that Joined Remotely, it checks whether the Instance is now Local to the DC and that the other necessary conditions are met (i.e. either the local ceiling has not been reached or the quorum test was run successfully) to remove the CPCM Instance from the list. It then updates the remote and local counts accordingly.

NOTE: If an Instance on the list of CICF that Joined Remotely is found to be no longer a member of the AD (i.e. the DC is informed by the Instance that it is either Blank or in a different AD), the DC can remove it from the list of CICF that Joined Remotely, decrement the remote count and increment the local count. This can occur in two situations: Firstly, this means the Instance has left with another DC and hence the Leave has been counted as Local. Secondly, the Instance may have been Blanked (e.g. reset to factory settings). In the latter case, one local count will be lost for the AD but this process recovers the remote count.

8.4.3.4 Local Master

The Local Master Security control has mainly the following roles during an AD Join Protocol:

- SAC establishment with the Blank Instance: If a SAC with the Blank Instance has not yet been established, it initiates the SAC establishment. It also accepts the SAC establishment request from the DC controller. Before establishing the SAC, it checks whether the AD capability indicated in the Begin message matches the one in the certificate, or at least, is compatible with the AD Join protocol.
- It determines then whether the protocol will be delegated or not: if the CPCM Instance is ADSE countable (determined from the Instance certificate obtained during the SAC establishment), the protocol is delegated.
- AD Secret delivery to the Blank Instance (whether the protocol is delegated or not): Upon receiving the AD Join Confirm message from a valid DC (if the protocol is delegated) or the AD Join Commit message from the Blank Instance, it sends the AD secret using the SAC.

8.4.3.5 Other AD members

Security Controls of other ADSE countable Instances may also be involved in the AD Joining protocol if the Domain Controller has to perform a Quorum Test in order to allow the Joining. In this case, they will first have to answer the quorum test and maybe afterwards respond to a proximity test. The latter test happens if the quorum was reached: for the test to be successful the DC also has to make sure the responding ADSE countable Instances are Local.

8.4.4 Instance Leaving the AD

AD Leave protocol is launched by user interaction either on the device of the CPCM Instance that has to leave or by ADM Invite. The protocol allows, when a CICF is leaving, to update the ADSE counts. Total count is incremented during the Leave protocol; the remote count is incremented if the Leaving Instance is in the DC's list of the CICFs that Joined Remotely, otherwise the local count is incremented. For a non-CICF Instance, running the protocol may be seen as unnecessary but it at least guarantees that there will be user confirmation at least on the device whose CPCM Instance is about to Leave or on the Local Master device.

8.4.4.1 Leaving Instance

The security control is first involved in the SAC establishment initiated by the LM (if the protocol is delegated) or a DC. The SAC needs to be renewed if it was already existing.

Before sending the AD Leave Commit message, the Security Control checks that the context verifies all the conditions (i.e. SAC established with a LM or DC, DC is a DC capable device that is a CICF). If all tests succeed and for a CICF, the CIC identifier of the DC controller with which the protocol is run is recorded in order to enable the protocol to resume if an error were to occur. The AD secret is disabled and therefore is no longer usable. While the Instance is no longer able to use its AD Secret to perform any AD related operation (AD challenge protocol, AD quorum test, content access), its status is still AD Member at that step.

It is then ready to delete the AD secret and ADID upon request of the Security Control of the LM or DC.

Finally, upon confirmation of the AD management (i.e. when AD Leave Confirm is received), it erases the stored CIC identifier, if any.

8.4.4.2 Domain Controller

There are two different circumstances under which a DC is involved in a Leave protocol:

- The protocol is run with a non ADSE countable Instance. In this case, the role of the DC is mostly the same as that of a simple Local Master: Upon reception of the AD Leave Begin message, it establishes a SAC with the Blank Instance and checks that the Leaving Instance is an AD member and that information transmitted in the AD Leave Begin message matches the one embedded in the certificate. Then, upon reception of the AD Leave Commit message, the Security Control asks the Leaving Instance to delete the AD Secret.

- The protocol is run with an ADSE countable Instance, in which case the role of the DC is fundamental as it has to check that all the conditions are met to update the ADSE counts.

The description below applies for an ADSE countable Instance where the C&R regime applies the standard ADSE method described in TS 102 825-7 [i.12].

When the DC receives an AD Leave Begin from a CICF, it first checks whether the requesting Instance is an AD member. It then establishes a SAC with the LM (if the message was delegated) or the requesting Instance, in which case it checks that the AD capability of the certificate matches the one of the message. If the original message was delegated but the DC knows how to address the Leaving Instance, it may decide to establish the SAC directly with this Instance and to stop using the delegation process. If the CICF is countable, it records the CIC identifier of the Leaving Instance.

Then, when the DC receives the AD Leave Commit message, the security control computes the future ADSE counts: Total count is incremented by one. Local count is also incremented by one, unless the Leaving Instance CIC identifier is in its list of CICFs that Joined Remotely, in which case remote count is incremented by one and the entry is removed from the list. These values are computed but are not yet enabled, awaiting confirmation from the Leaving Instance. It then requests the Leaving Instance to erase the AD secret or asks the LM to proceed in delegation mode.

Finally, upon receiving the AD Leave Finish message, the DC knows the protocol was successful and requests the security control to erase the CPCM instance certificate identifier. It then enables the new ADSE values.

8.4.4.3 Local Master

The Local Master Security control has mainly the following roles during an AD Leave Protocol:

- SAC establishment with the Leaving Instance: It initiates the SAC establishment even if a SAC was already existing, in which case it is renewed. It also accepts the SAC establishment request from the DC controller. Before establishing the SAC, it checks whether the Leaving Instance is from the correct AD and that information transmitted in the AD Leave Begin message matches the Leaving Instance certificate.

- It then determines whether the protocol will be delegated or not: if the CPCM Instance is ADSE countable (determined from the Instance certificate obtained during the SAC establishment), the protocol is delegated.

- It sends an AD Secret erasure message to the Leaving Instance, upon receiving the AD Leave Confirm message from a valid DC (if the protocol is delegated) or the AD Leave Commit message from the Leaving Instance.

8.4.5 Domain Controller Transfer

8.4.5.1 AD member becoming DC

The security control is first involved in the SAC establishment initiated by the DC that will be transferred. The SAC is renewed if it was already established. It also replies to any proximity test performed by the DC to check whether the Transfer is Remote or Local.

Before sending the DC Transfer Commit message, the Security Control checks that the context verifies all the conditions (i.e. SAC established with a DC, DC is a DC capable device that is a CICF; Instance is DC capable and a CICF). If all tests succeed, the CIC identifier of the DC controller, with which the protocol is run, is recorded in order to enable the protocol to resume if an error were to occur.

It is then ready to receive the information necessary to become a DC (ADSE counts, list of CICF that Joined Remotely…)

Finally, upon confirmation from the AD management (i.e. when DC Transfer Confirm is received), it becomes the DC and erases the stored CIC identifier. It looks in the received list of CICFs that Joined remotely to determine whether it is present, and, in such a situation, it will try to remove itself from this list (as it is Local to itself), which will be possible if the Local count has not reached local ceiling or if the quorum test passes.

8.4.5.2 Domain Controller Transfer Process

When the DC receives a DC Transfer Begin from an AD member, it first checks whether the requesting Instance is an AD member, that is DC capable and that it is a CICF. It also performs a Proximity test with the requesting Instance to check whether the Transfer will be Remote or not. If it is remote, it checks whether the DC remote count has reached its ceiling (in this case, it may try to run ADM AAA protocol). It then establishes a SAC with the requesting Instance, and checks that the AD capability of the certificate matches the one of the message.

Then, when the DC receives the DC Transfer commit message, the security control computes the future ADSE counts if the transfer is remote. It disables all the counts, and will thus no longer be capable of performing other DC related protocols. It ceases to be a DC. It also stores the CIC identifier of the future DC. It then transfers to the requesting Instance the DC capacity and the ADSE counts and the list of CICFs that Joined remotely.

Finally, upon receiving the DC Transfer finish message, the DC knows that the protocol was successful and requests the security control to erase the CPCM instance certificate identifier. It also erases ADSE counts and the list of CICFs that Joined Remotely.

8.4.6 Domain Controller Splitting

8.4.6.1 AD member becoming an additional DC

The security control is first involved in the SAC establishment initiated by the splitting DC. The SAC needs to be renewed if it was already established. It also replies to any proximity test performed by the DC to check whether the Split is Remote or Local.

Before sending the DC Split Commit message, the Security Control checks that the context verifies all the conditions (i.e. SAC established with a DC, DC is a DC capable device that is a CICF; Instance is DC capable and a CICF). If all tests succeed, the CIC identifier of the DC controller, with which the protocol is run, is recorded in order to enable the protocol to resume if an error were to occur. It also checks that the requested ADSE values are consistent with the current values in the DC or will not cause any inconsistency after the Split.

It is then ready to receive the information that is necessary to become a DC (ADSE counts, list of CICF that Joined Remotely…).

Finally, upon confirmation from the AD management (i.e. when DC Split Confirm is received), it becomes the DC and erases the stored CIC identifier.

8.4.6.2 Domain Controller in Splitting Process

When the DC receives a DC Split Begin from an AD member, it first checks whether the requesting Instance is an AD member, that it is DC capable and that it is a CICF. It also performs a Proximity test with the requesting Instance to check whether the Transfer will be Remote or not. If it is remote, it checks whether the DC remote count has reached its ceiling (in this case, it may try to run ADM AAA protocol). In addition, it checks whether the DC split ceiling has not been reached (ADMAAA may also be contacted in the contrary case). It then establishes a SAC with the requesting Instance, and checks that the AD capability of the certificate matches the one of the message.

Then, when the DC receives the DC Split commit message, the security control checks that the requested ADSE values and the list of CICFs that Joined Remotely are consistent with each other. It computes the future ADSE counts of each DC, but these counts are not yet enabled. It also updates the list of CICFs that Joined Remotely if some identifiers were requested. If the requesting Instance is in the list, its identifier will be automatically sent (and thus also transfers a remote count of at least one) so that the additional DC may gain a remote slot.

It also stores the CIC identifier of the future DC. It then transfers to the requesting Instance the DC capacity, its ADSE counts and a list of CICF that Joined remotely, if requested.

Finally, upon receiving the DC Split finish message, the DC knows the protocol was successful and requests the security control to erase the CPCM instance certificate identifier. It then enables its new ADSE counts.

8.4.7 Domain Controller Merging

8.4.7.1 Merging Domain Controller

The security control is first involved in the SAC establishment initiated by the merging DC. The SAC needs to be renewed if it was already established.

Before sending the DC Merge Commit message, the Security Control checks that the context verifies that all the conditions (i.e. SAC established with a DC, DC is a DC capable device that is a CICF; Instance is DC capable and a CICF) are met. If all the tests succeed, the CIC identifier of the DC controller with which the protocol is run, is then recorded in order to enable the protocol to resume if an error were to occur. In addition, it will no longer be able to run any other DC related protocol even though it will still be discovered as a DC.

It will then be ready to provide the other DC with its information (ADSE counts, list of CICF that Joined Remotely…).

Finally, upon confirmation from the AD management (i.e. when DC Merge Confirm is received), it updates its ADSE counts to integrate what it had received and then erases the recorded CIC identifier. It also totally ceases to be a DC.

8.4.7.2 Merged Domain Controller

When the DC receives message DC Merge Begin from another DC, it first checks whether the requesting Instance is a DC from the same AD. It then establishes a SAC with the requesting Instance, and checks that the AD capability of the certificate matches that of the message.

Then, when the DC receives the DC Merge Commit message, the security control computes the future ADSE counts of the DC, and checks their consistency, but these counts are not yet enabled. It stores the transferred CICFs identifiers. It increments the DC split count by one. It also stores the CIC identifier of the merged DC and informs the other DC that the merge can proceed. If the other DC confirms, it informs its ADM control that message DC Merge Confirm can be sent.

Finally, upon receiving the DC Merge finish message, the DC knows that the protocol was successful and requests the security control to erase the CPCM instance certificate identifier. It enables its new ADSE counts and the new list of CICFs that Joined Remotely.

8.4.8 Domain Controller Rebalancing

8.4.8.1 Rebalancing Domain Controller

The security control is first involved in the SAC establishment initiated by the rebalancing DC. The SAC needs to be renewed if it was already established.

Before sending the DC Rebalance Commit message, the Security Control checks that the context verifies all the conditions (i.e. SAC established with a DC, DC is a DC capable device that is a CICF). It also checks the consistency of the request (i.e. total_count is the sum of remote_count and local_count, final counts will be lower than final ceilings, requested counts are lower than or equal to the ones of the peer DC, requested CICF identifiers are in the list of CICFs that Joined Remotely to the peer DC). If all the tests succeed, the CIC identifier of the DC controller with which the protocol is run, is then recorded in order to enable the protocol to resume if an error were to occur.

It will then receive the ADSE counts and the CICF identifiers from the other DC. These values are stored but are not yet enabled.

Finally, upon confirmation from the AD management (i.e. when DC Rebalance Confirm is received), it updates its ADSE counts to integrate what it had received and then erases the recorded CIC identifier.

8.4.8.2 Rebalanced Domain Controller

When the DC receives message DC Rebalance Begin from another DC, it first checks whether the requesting Instance is a DC from the same AD. It additionally checks that the DC split count has not reached its ceiling. It then establishes a SAC with the requesting Instance, and checks that the AD capability of the certificate matches the one of the message.

Then, when the DC receives the DC Rebalance Commit message, the security control checks that the requested ADSE values and list of CICFs that Joined Remotely are consistent. It computes the future ADSE counts of the DC and checks their consistency. These counts are immediately enabled, but previous ones are kept in order to enable the protocol to resume. It removes from its list of CICFs that Joined Remotely those identifiers to be transferred. Again, these values are to be kept in order to enable the protocol to resume.

It stores the CIC identifier of the rebalanced DC and sends message Rebalance DC. If the other DC confirms, it informs its ADM component that it can send message DC Rebalance confirm.

Finally, upon receiving the DC Rebalance finish message, the DC knows that the protocol was successful and requests the security control to erase the CPCM instance certificate identifier. It erases the previous ADSE counts and the transferred identifiers.

8.5 Generic ADSE Tools usage

This multi-part deliverable provides a set of tools upon which an ADSE method can be designed. All these tools are exemplary and hence implementations do not need to support them, unless required by the applicable C&R regime.

C&R regimes can use these tools (or any other ones, the list being not exhaustive) to build another ADSE method. In this case, it is not recommended to use one single tool but to use a combination of them. Each tool actually addresses only part of the expectations of an ADSE method and would result in an unbalanced ADSE method.

8.5.1 Single Household Metric

Single Household Metric (SHM) is not a tool as such but rather a mandatory ingredient in any ADSE method recipe. The tool aims actually to ensure that the AD spans over one single household and is not shared by several of them.

Examples of SHM tool include:

- Limiting the number of separate LAN clusters in the AD. A single house has often actually no more than 3 distinct clusters (the main house, the car and the second house), if mobile devices are not taken into account. However, including support for mobile devices makes this kind of heuristic harder to use.

- Limited number of simultaneous viewings in an AD. The rationale is to say that since a family has a limited number of members, there is the same limit on simultaneous displays in the house. This allows for not imposing any limit on the number of devices in the house. While attractive, this method is hard to set-up since it is difficult to estimate what the actual number of simultaneous viewings (content transferred for storage or processing will not be counted) might be, especially in the case where the AD has several disconnected, and often unconnected, clusters.

- Limits on the number of devices. This tool is the easiest to set up but has some limitations. If the limit is set too low, this may prevent a single household from having all of its devices being included in the same AD while if it is set too high, it makes the sharing of a single AD between multiple households possible. Refinements of such approach are described in clauses 8.5.2 to 8.5.8.

8.5.2 Total AD Count

Total AD count consists of limiting the total number of CPCM Instances in an AD. If not used in conjunction with other methods, this tool has the following drawbacks:

- No distinction is made between Remote and Local AD Joins. Hence, the AD could consist of a set of CPCM Instances that are all Remote from each other. This would no longer match the definition of an AD and could be use as a circumvention tool.

- Counting all the CPCM Instances does not necessarily reflect actual AD capabilities. As an example, a Storage Entity of 1 GB counts as one while 10 Storage Entities of 100 MB count as 10, while the user has exactly the same storage capabilities.

Total AD count supposes that a single Instance in the AD keeps a count of Instances that Joined and Leaved. This CPCM Instance is a Domain Controller. It is then recommended to allow for the Domain Controller function to be Transferred so as to allow the user to select which of his devices will play this role.

8.5.3 Total and Remote AD count

Total and Remote AD count (TARC) consists of limiting the total number of CPCM Instances in an AD and the number of CPCM Instances that may Join Remotely. Hence, the TARC tool addresses one of the limitations of the Total AD count tool.

When the TARC tool is used, it is also recommended to allow for the Domain Controller to be Split. This permits the user to have a Domain Controller per AD cluster. The number of Remote Joins can thus be set very low to control the number of clusters, while allowing Local Join in each cluster. If the Split is implemented, the Merge Protocol, which allows the merging of two Domain Controllers, is generally also implemented. When a Domain Controller function is Split, the total and remote ceilings are also split between the two Domain Controllers. It is hence also recommended to allow the user to rebalance these counts between the two Domain Controllers, which is the purpose of the Rebalance protocol.

When TARC tool is used, the capacity of Remotely Transferring Domain Controller needs to be limited. Else, limiting the Remote Joining capability would be easy to circumvent as the user could Transfer its Domain Controller to the Remote location, let the new CPCM Instance Join Locally and re-Transfer the Domain Controller to the original location.

Similarly, if the Split is implemented, the number of Splits will be also limited as the number of Domain Controllers should not be much higher than the number of expected clusters in an AD.

When the TARC tool is used, a recommended additional feature is for DCs to detect when a CPCM Instance that Joined Remotely is Local so that they can consider that the corresponding Instance Joined Locally. This feature allows a user to Join newly purchased devices as soon as they are purchased (i.e. probably remotely) while impacting the remote count only until when the devices are brought back home, at which point in time the devices will no longer be counted as Remote. To implement that feature, each DC has to maintain a list of CPCM Instances that Joined remotely. When detecting that a CPCM Instance in this list is connected and Local, the DC will remove the entry from the list, and decrease by one the count of Remote Joins (total count being not affected). It is also recommended to allow for the list of CPCM Instances that Joined Remotely to be also Transferred, Split, Merged and Rebalanced. This allows for counting as Local, an Instance that Joined Remotely and to which the DC is remotely Split. Finally, it also allows to count as Remote Leave those CPCM Instances that Joined Remotely (this cannot be done if there is no such list, as this could give means to circumvent the remote ceiling by letting Instances Join Locally and then Leave Remotely; hence without such a list, a Leave is necessarily seen as Local).

8.5.4 Total and Remote AD count +

Total and Remote AD count + (TARC+) method is an extension of TARC method where the remote_ceiling limit can be bypassed if another condition is met, typically the number of AD members that are connected at the time of the Joining.

The Quorum Test, described in next clause, is an example of TARC+ method. Another example would be a fixed number of AD members that need to be connected.

8.5.5 Quorum Test

As noted before, CPCM was designed to "just work" for the vast majority of legitimate users. The quorum approach was therefore included to support those consumers who have large numbers of devices in their legitimate household. Consumers who purchase and use many devices also tend to purchase lots of content, so they are good customers for the whole value chain. The quorum approach allows many devices to join the domain, as the local presence of the other devices gives a very strong indication that this is a legitimate household network, even if a large one. The exact parameters of such a domain are of course determined by the governing C&R Regime.

The Quorum Test is always used with the TARC method. When it is used, a local ceiling is set up under which local Joins are authorized irrespective of the count of connected CPCM Instances. Above that ceiling, a percentage of AD members have to be present to allow the Join. This percentage is always computed against current counts of the AD and not any ceilings, as this would lead to a hard limit (and hence to the TARC+ tool). The percentage may be computed against:

- The total count of AD members, the available Instances being Local or Remote. This case is a bit tough to set-up for the user as it needs to have a connection with as many Remote locations as possible.
- The total count of AD members, the available Instances being Local only. While the above problem has disappeared, this configuration might be hard to understand as it depends upon Instances that are out of its sight and which have never been Local and which will vary after a Remote Join. It may thus cause a bad user experience as future Joins would depend on making Local those Instances that have never been Local before. If this configuration is chosen, the C&R regime has to be careful to set the quorum percentage sufficiently to allow for a Join when the local ceiling has just been reached while all the Remote Joins have been done.
- The Local count of AD members, the available Instances being Local only. This configuration does not suffer from the drawbacks of the above configurations and allows for setting a quorum percentage which is relatively high. Only mobile devices may be absent and they can easily be made available if needed.

When the Quorum Test is used, it is recommended to allow an AD to Join on the Remote count if the local ceiling has been exceeded and the Quorum Test has failed. This would be actually nonsense from a security point of view to allow only Remote Joins.

If the Quorum Test is used and the TARC tool allows to re-count as Local an Instance that would have Joined Remotely but is now local, this re-balancing will occur under the same conditions as a Local Join: automatically if the local ceiling has not been reached and upon a successful quorum test otherwise.

8.5.6 Domain Membership History

This tool aims at limiting the number of ADs a given CPCM Instance may Join during its lifetime. The goal is to mitigate the threat where a CPCM Instance would temporarily Join (probably Remotely) an AD in order to access a CPCM content that would be bound to this AD and Leave afterwards. In this way, all the content that is marked MAD or VAD would be easily sharable with any CPCM Instance while temporarily affecting the ADSE counts.

Using the Domain Membership History tool, the above scenario may only be performed a limited number of times as the history count will be incremented by one each time an external AD is Joined (and by two if the CPCM Instance re-Joins its original AD in the meantime).

NOTE: When this tool is used, a different limit may be set for rental devices that will by definition Join more ADs than other devices. An alternative behaviour is to allow the rental company to re-initialize the Domain Membership History ceiling so that the counter can be set to a low value for each rental.

8.5.7 Wayfaring Device Limits

The goal of the Wayfaring Device Limits tool is to force a regular contact (i.e. running a SAC establishment protocol) between each AD member and the Domain Controller. A limit may be also set up to force Local connection to the Domain Controller. The goal of such a tool is to prevent a user letting a CPCM Instance Join Locally and then distributing (or re-selling) the device in order to re-distribute protected content.

If used in conjunction with the ability to Split Domain Controllers, this tool provides a very good countermeasure against the above threat while not inconveniencing honest users. Sedentary devices will actually always be in contact with the Domain Controller and will not be affected. Mobile devices will be regularly disconnected from the Domain Controller but should also be regularly connected.

The main drawback of this tool is that it forces each AD-capable device to be relative or absolute time aware to measure whether its AD membership is maintained or not. As the time limit is supposed to be relatively long (e.g. to allow the user having a 3 weeks vacation period), this will increase the need for implementing secure clocks in mobile devices and having a permanent access to a secure clock in each AD cluster. Thus, the cost of mobile devices is likely to be highly impacted.

8.5.8 ADM by AAA

Whatever the flexibility of the ADSE method selected by a CPCM C&R regime is, there will always be corner case situations where an honest user will be denied a legitimate Join. ADMAAA tool allows for contacting an ADMAAA which may overcome the reason that the ADM operation was refused, and subsequently authorize it. This authorization may be valid once or it may define new values for ADSE ceilings so that the user may continue to manage its AD while not being concerned with contacting the ADMAAA each time.

While two messages only are defined for this tool, a given ADM AAA may define additional messages to confirm the validity of the DC request. Similarly, the way an ADM AAA is authenticated as an AAA will be defined by the ADM AAA. This could be for instance by having a dedicated certificate identifier which is set in each AD-aware CPCM Instance at the time of manufacture.

ADM AAA may also be used as a single tool for AD management. While this choice probably gives more flexibility for the user and more confidence for the content owner, it does not provide any more privacy for the user.

8.6 ADM Messaging

8.6.1 General Messaging

CPCM devices within the home network exchange messages to administer the Authorised Domain. The patterns of message exchange are a direct function of the role of the CPCM device for general messaging. There are four different role types which are as follows:

BI	Blank Instance: The device is not a member of any AD.
DM	AD Member: The device is a member of an AD.
LM	Local Master: The Local Master is the single point of control within the home network. The other devices in the home network transmit requests to the Local Master to participate in the AD. If the Local Master does not control the AD itself, it forwards the messages to another (possibly remote) device that does control the AD.
DC	Domain Controller: The device administers the AD. A device that implements this role decides upon AD membership and enforces membership obligations for CICF.

8.6.2 Message Exchange Patterns

Table 5 presents the message exchange patterns.

Table 4: ADM Message Exchange Patterns

Message	Device	Source	Sink	Delegatable
AD Change Request	unicast	DM/LM	LM/DC	Yes
AD Change Response	unicast	LM/DC	DM/LM	Yes
AD Join Begin	unicast or broadcast	BI/LM	LM/DC	Yes
AD Join Commit	unicast	BI/LM	LM/DC	Yes
AD Join Confirm	unicast	LM/DC	BI/LM	Yes
AD Join Finish	unicast	DM/LM	LM/DC	Yes
AD Join Ready	unicast	LM/DC	BI/LM	Yes
AD Leave Begin	unicast	DM/LM	LM/DC	Yes
AD Leave Commit	unicast	DM/LM	LM/DC	Yes
AD Leave Confirm	unicast	DC/LM	LM/DM	Yes
AD Leave Finish	unicast	BI/LM	LM/DC	Yes
AD Leave Ready	unicast	DC/LM	DM/LM	Yes
AD Update Indication	broadcast	LM/DC	LM/DM	No
AD Update Request	unicast	DM	LM	No
AD Update Response	unicast	LM	DM	No
ADM Invite	unicast	Any	Any	No
ADM Invite Result	unicast	Any	Any	No
ADM quorum test query	broadcast	DC	DC/DM	No
ADM quorum test response	unicast	DC/DM	DC	No
ADMAAA tool request	unicast	DC	ADMAAA	No
ADMAAA tool response	unicast	ADMAAA	DC	No
Become Domain Controller	unicast	DM/DC	DM	No
Deliver AD secret	unicast	DC/LM	BI	No
Deliver AD secret response	unicast	BI	DC/LM	No
Domain Controller Merge Begin	unicast	DM	DC	No
Domain Controller Merge Commit	unicast	DM	DC	No
Domain Controller Merge Confirm	unicast	DC	DM	No
Domain Controller Merge Finish	unicast	DC	DC	No
Domain Controller Merge Ready	unicast	DC	LM	No
Domain Controller Merged	unicast	DM	DC	No
Domain Controller Rebalance Begin	unicast	DC	DC	No
Domain Controller Rebalance Commit	unicast	DC	DC	No
Domain Controller Rebalance Confirm	unicast	DC	DC	No
Domain Controller Rebalance Finish	unicast	DC	DC	No
Domain Controller Rebalance Ready	unicast	DC	DC	No
Domain Controller Rebalanced	unicast	DC	DC	No
Domain Controller Split Begin	unicast	DM	DC	No
Domain Controller Split Commit	unicast	DM	DC	No
Domain Controller Split Confirm	unicast	DC	DM	No
Domain Controller Split Finish	unicast	DC	DC	No
Domain Controller Split Ready	unicast	DC	DM	No
Domain Controller Transfer Begin	unicast	DM	DC	No
Domain Controller Transfer Commit	unicast	DM	DC	No
Domain Controller Transfer Confirm	unicast	DM	DM	No
Domain Controller Transfer Finish	unicast	DC	DM	No
Domain Controller Transfer Ready	unicast	DC	DM	No
Erase AD secret	unicast	DC	DM/LM	No
Erase AD secret response	unicast	DM/LM	DC	No
Local Master election indication	broadcast	LM/DC	DM/LM	No
Local Master election request	broadcast	DM	DM	No
Local Master election response	broadcast	DM	DM	No
Merge Domain Controller	unicast	DC	DM	No
New Domain Controller	unicast	DM	DM	No
Rebalance Domain Controller	unicast	DC	DC	No

NOTE: In this table, only roles played in the course of a protocol are shown. This does not exclude the possibility that a LM or a DC acts as a simple DM during a protocol.

8.6.3 Interfaces Definition

The message exchange patterns above are allocated to three domain interfaces:

a) AD member capable.

b) Local Master capable, and c) Domain Controller capable.

8.6.3.1 AD Member Interface

This interface transmits and receives messages for the `local_master_election`, `AD_update`, `AD_leave`, `AD_join` and `AD_change` functions. The table 6 lists the requisite message exchange patterns.

Table 5: AD Member Message Exchange Patterns

Message	Mode	Transmit	Receive
AD_change_request	unicast	X	---
AD_change_response	unicast	---	X
AD_Join_Begin	unicast	X	---
AD_Join_Commit	unicast	X	---
AD_Join_Confirm	unicast	---	X
AD_Join_Finish	unicast	X	---
AD_Join_Ready	unicast	---	X
AD_Leave_Begin	unicast	X	---
AD_Leave_Commit	unicast	X	---
AD_Leave_Confirm	unicast	---	X
AD_Leave_Finish	unicast	X	---
AD_Leave_Ready	unicast	---	X
AD_quorum_test_query	broadcast	---	X
AD_quorum_test_query_response	unicast	X	---
AD_update_indication	broadcast	---	X
AD_update_request	unicast	X	---
AD_update_response	unicast	---	X
ADM Invite	unicast	X	X
ADM Invite Response	unicast	X	X
Deliver_AD_secret	unicast	---	X
Deliver_AD_secret_response	unicast	X	---
Erase_AD_secret	unicast	---	X
Erase_AD_secret_response	unicast	X	---
Local_Master_election_indication	broadcast	---	X
Local_Master_election_request	broadcast	---	X
Local_Master_election_response	broadcast	X	---

8.6.3.2 Local_Master Interface

The Local Master interface extends the AD Member interface. The table below presents the requisite message exchanges.

Table 6: Local Master Message Exchange Patterns

Message	Mode	Transmit	Receive
AD_change_request	unicast	X	X
AD_change_response	unicast	X	X
AD_Join_Begin	unicast	X	X
AD_Join_Commit	unicast	X	X
AD_Join_Confirm	unicast	X	X
AD_Join_Finish	unicast	X	X
AD_Join_Ready	unicast	X	X
AD_Leave_Begin	unicast	X	X
AD_Leave_Commit	unicast	X	X
AD_Leave_Confirm	unicast	X	X
AD_Leave_Finish	unicast	X	X
AD_Leave_Ready	unicast	X	X
AD_Quorum_Test_Query	broadcast	---	X
AD_Quorum_Test_Response	unicast	X	---
AD_update_indication	broadcast	X	X
AD_update_request	unicast	X	X
AD_update_response	unicast	X	X
ADM_invite	unicast	X	X
ADM_invite_response	unicast	X	X
Deliver_AD_Secret	unicast	X	X
Deliver_AD_Secret_response	unicast	X	X
Erase_AD_Secret	unicast	X	X
Erase_AD_Secret_response	unicast	X	X
Local_Master_election_indication	broadcast	X	X
Local_Master_election_request	broadcast	X	X
Local_Master_election_response	broadcast	X	X

8.6.3.3 Domain_Controller Interface

The Domain Controller interface extends the Local Master interface. In addition to the functions of the Local Master interface, the interface implements the DC Transfer, Split, Merge and Rebalance functions. Table 8 presents the requisite message exchanges.

Table 7: Domain Controller Message Exchange Patterns

Message	Mode	Transmit	Receive
AD_Change_Request	unicast	X	X
AD_Change_Response	unicast	X	X
AD_Join_Begin	unicast	X	X
AD_Join_Commit	unicast	X	X
AD_Join_Confirm	unicast	X	X
AD_Join_Finish	unicast	X	X
AD_Join_Ready	unicast	X	X
AD_Leave_Begin	unicast	X	X
AD_Leave_Commit	unicast	X	X
AD_Leave_Confirm	unicast	X	X
AD_Leave_Finish	unicast	X	X
AD_Leave_Ready	unicast	X	X
AD_Quorum_Test_Query	broadcast	X	X
AD_Quorum_Test_Response	unicast	X	X
AD_update_indication	broadcast	X	X
AD_update_request	unicast	X	X
AD_update_response	unicast	X	X
ADM_invite	unicast	X	X
ADM_invite_response	unicast	X	X
ADMAAA_Tool_Request	unicast	X	---
ADMAAA_Tool_Response	unicast	---	X
Become_Domain_Controller	unicast	X	X
DC_Merge_Begin	unicast	X	X
DC_Merge_Commit	unicast	X	X
DC_Merge_Confirm	unicast	X	X
DC_Merge_Finish	unicast	X	X
DC_Merge_Ready	unicast	X	X
DC_Rebalance_Begin	unicast	X	X
DC_Rebalance_Commit	unicast	X	X
DC_Rebalance_Confirm	unicast	X	X
DC_Rebalance_Finish	unicast	X	X
DC_Rebalance_Ready	unicast	X	X
DC_Split_Begin	unicast	X	X
DC_Split_Commit	unicast	X	X
DC_Split_Confirm	unicast	X	X
DC_Split_Finish	unicast	X	X
DC_Split_Ready	unicast	X	X
DC_Transfer_Begin	unicast	X	X
DC_Transfer_Commit	unicast	X	X
DC_Transfer_Confirm	unicast	X	X
DC_Transfer_Finish	unicast	X	X
DC_Transfer_Ready	unicast	X	X
Deliver_AD_Secret	unicast	X	X
Deliver_AD_Secret_response	unicast	X	X
Domain_Controller_Merged	unicast	X	X
Domain_Controller_Rebalanced	unicast	X	X
Erase_AD_Secret	unicast	X	X
Erase_AD_Secret_response	unicast	X	X
Local_Master_election_indication	broadcast	X	X
Local_Master_election_request	broadcast	X	X
Local_Master_election_response	broadcast	X	X
Merge_Domain_Controller	unicast	X	X
New_Domain_Controller	unicast	X	X
Rebalance_Domain_Controller	unicast	X	X

8.6.4 Local Master Delegation

Delegation is the mechanism by which any CPCM Instance can discover and communicate with a Domain Controller, even if it is Remote and thus not directly discoverable. The idea is for the CPCM Instance to contact the Local Master, which has the capability to discover the available Domain Controllers. The Local Master will act as a proxy to transmit messages between the Domain Controller and the CPCM Instance. In addition, the Security Control of the Local Master also acts as a proxy for the Security Control of the Domain Controller to deliver the AD Secret (in case of Joining) or to check the AD Secret has been erased (in case of Leaving).

The delegation mechanism is only used for the AD Join or the AD Leave Protocols that may involve low-capability devices. All other protocols requiring the presence of a Domain Controller – DC Transfer, Split, Merge or Rebalance - necessarily involve a DC capable CPCM Instance that is able to directly contact the DC to run the protocol.

When starting the protocol, the CPCM Instance about to Join or to Leave the AD contacts the Local Master which then determines with which DC (if any available) the protocol is run. It then forwards the message to that DC, including in the message the CIC identifier of the requesting CPCM Instance so that the DC knows with whom the protocol is actually being run. This is required to resume the protocol in case of any problems (see clause 6.3.3.4). Then, all other protocol messages exchanged between the LM and the DC will include that same CIC identifier. This allows the LM to know which CPCM Instance the message from the DC needs to be redirected to and the DC to identify which protocol it is running such as in the case when it is capable of running simultaneous Join Protocols. Similarly, all messages exchanged between the LM and the requesting CPCM Instance will carry the DC CIC identifier so that the requesting Instance knows with which DC the protocol is run and the Local Master knows to which DC it needs to redirect the messages.

The Local Master only makes two kinds of modifications to the messages that it forwards:

- delegating_instance_id field which is updated as explained above;
- security related: When applicable, the LM removes the SAC protection of the incoming message to put different SAC protection, which uses a different SAC session key, to the updated outgoing message. Even if no field was updated, this modification is still necessary as the SAC keys are not the same.

All other message elements are unmodified so that they carry the information of the actual CPCM Instances that are involved in the protocol (and no information about the Local Master capabilities).

8.7 "Marriage and Divorce" in CPCM

The CPCM Authorised Domain provides a means to associate the devices owned, rented, or otherwise controlled by a single household. However, in the real world, households are dynamic entities which can be combined and split. While the CPCM specifications do not directly support this in the Authorised Domain, it is possible to implement products that can handle these situations in an acceptable manner.

8.7.1 Migration

These guidelines use the term "migration" to refer to the process of moving something from one logical Authorised Domain to another. Both devices (i.e. CPCM Instances) and content (CPCM Content Licenses) can be migrated when correctly authorised.

Migration of a device is achieved using the normal ADM protocols (AD Leave and AD Join), or by proprietary means in the case of AD-aware CPCM Instances.

Migration of content is more complex, and may require the use of special equipment as described below. Some content may be impossible to migrate in any case.

It is not necessary to descramble and rescramble the actual audio-visual content. Migration can be implemented by updating the ADID in the Content License and, when the CL is protected with the AD secret, decrypting the CL with the original AD Secret and re-encrypting it using the new AD Secret.

8.7.2 New ADM Processes.

There are two ADM processes described here.

- Combining two Authorised Domains into a single Authorised Domain (Marriage); and
- Separating a single Authorised Domain to two or more Authorised Domains (Divorce).

In each case, content needs to be migrated between domains, something which is not always permitted in a CPCM system. This therefore requires implementation of a device specific to this purpose. It is anticipated that such a capability is limited to use by specific professionals, such as authorised agents of the C&R Regime, who are able to ensure that the process takes place within an appropriate legal framework.

It is important to note that these processes are agreed by the governing C&R Regime.

8.7.3 AD Conversion Device

An AD Conversion Device is a special purpose unit specifically designed to handle the migration of content between Authorised Domains. The architecture of the device is as follows.

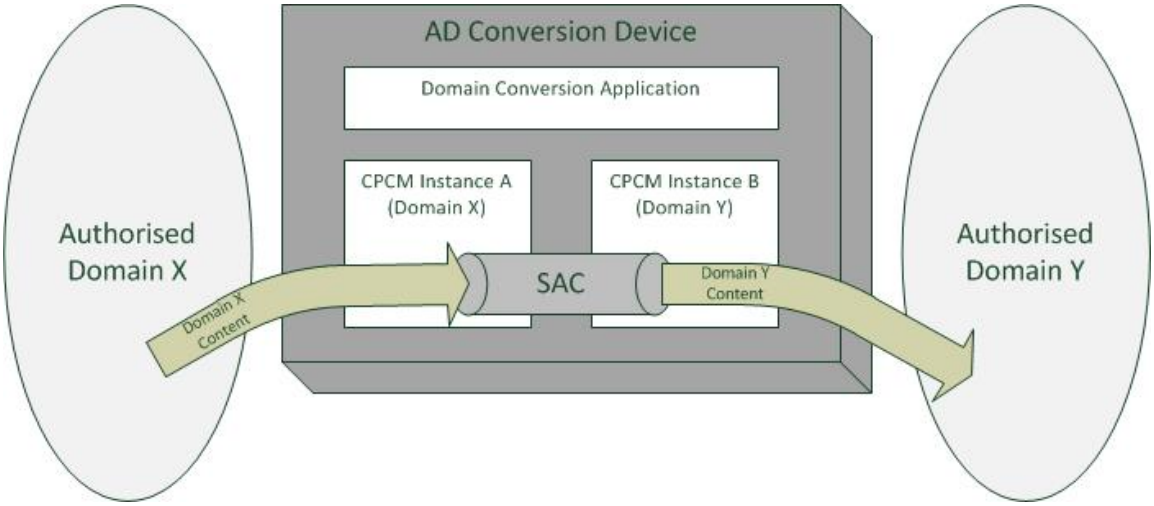


Figure 21: AD Conversion Device

The AD Conversion device includes two separate CPCM Instances. One CPCM Instance is Joined (using normal ADM protocols) to the source AD, while the other is Joined to the destination AD.

A SAC is established between the two CPCM instances within the AD Conversion Device, allowing them to communicate securely.

Both CPCM Instances in the AD Conversion Device need to be DC-capable, as they will need to create and destroy ADs.

The two CPCM Instances in the device are always in Proximity (inherently by design) and do not need to run protocols to test for proximity with each other.

Ideally, all devices that belong to both domains (including any bit-bucket stores holding CPCM content) should be present, connected, and discoverable, though this is not absolutely essential.

Domain-bound CPCM Content Licenses, with or without their associated content, are passed to the Receiving Instance; the CPCM Instance A in the diagram, of the AD Conversion Device by devices in the source AD. If the CL is protected by the AD Secret, the Receiving Instance decrypts the received CPCM Content License using the source AD Secret, and then passes it securely to the Sending Instance; the CPCM Instance B in the diagram. The Sending Instance updates the Content License with the new ADID, and protects the new Content License using the mechanisms described in clause 6.5.6. The Content License is then sent to a device in the destination AD.

The AD Conversion Device needs some special exemptions from the governing C&R Regime:

- Firstly it is authorised to move content between ADs, even if the content is not marked as MLocal or MCPCM.
- Secondly, the AD Conversion Device needs to be completely unconstrained as to the number of domains it is permitted to create, Join, or leave over its lifetime.
- Thirdly, both CPCM Instances within the AD Conversion Device are treated as non-ADSE-countable (non-CICF), so as to allow them to Join a domain even if it is already "full", and without affecting the ADSE counts.

NOTE: The third exemption overrides the usual rule that a Domain Controller function needs to reside within an ADSE-countable CPCM Instance.

For these reasons, the C&R Regime is likely to closely control the ownership and use of such devices to known and trusted individuals or organisations.

8.7.4 Combining Authorised Domains (Marriage)

In this case, such as when two people with separate CPCM ADs decide to form a combined household, there is a need to migrate content from one AD to the other. It is recommended that the "larger" domain is retained, and the devices and content bound to the small domain are migrated into it.

Content is moved to the chosen destination AD using the AD Conversion Device as described above.

Once all content has been moved (including any content stored on bit-buckets), the devices in the source domain should all be told to Leave the domain (using normal ADM protocols, perhaps initiated by an AD Leave Invitation message sent from the AD Conversion Device). As each device completes the Leave protocol, it can start the Join protocol with the destination domain.

It is important that all devices belonging to the source domain are known to have left the domain. Ideally this will include all devices. However it is adequate if just the ADSE-countable devices all Leave. Some C&R Regimes may be satisfied with some devices being left, provided that no DCs remain, as this will prevent any later expansion of the (hopefully) abandoned domain.

NOTE: In practice it is not possible to verify that all non-ADSE-countable devices Leave, as there is no record kept of them ever joining in the first place.

Once all the devices have completed their Leave protocol, the only member of the source AD should be the Receiving CPCM Instance inside the AD Conversion Device. This device can now Leave the domain, which will cease to exist. Any stray content and/or content licenses that may still exist on other media will now be useless and can be ignored.

To summarise, the AD Conversion Device is first used to migrate content from one domain into the other. Then it is used to assist in moving the devices to their new domain, and finally in ensuring the deletion of the redundant domain.

8.7.5 Splitting an Authorised Domain (Divorce)

Dividing a single AD into two or more domains is necessary when a household becomes divided, such as when a child leaves home or when a couple decide to separate.

For the purpose of this process, a new AD is created and a proportion of the devices and content are moved to this new AD, while others will remain as they are.

The rules for domain-bound content require it can only be bound to a single domain, unless it is marked as MLocal or MCPCM. It is therefore necessary for a decision to be made for each such content item as to whether it will remain in the original domain or migrate to the new domain.

Because the new AD is created, it will get a new set of ADSE counts independent of the ADSE counts for the original AD.

The process is as follows:

- The AD Conversion Device application joins one of its CPCM Instances to the original domain, using normal ADM protocols as described above.
- The AD Conversion Device application tells the other of its CPCM Instances to create a new domain, (using the normal ADM process). This CPCM Instance will thus become the DC and LM for the new domain as usual.
- One or more devices with Storage Entity capability are migrated to the new domain by Leaving the original domain and Joining the new one. Or they may just Join if they are new devices bought specifically for the purpose of holding the migrated content.
- The AD Conversion Device is used to migrate the Content Licenses of all content that will be moved to the new domain. For each domain-bound content item, a decision needs to be made as to whether to migrate it wholly to the new domain or leave it wholly in the original domain. See below for some special considerations when such content is marked as Copy-Control-Not-Asserted.

Both domains continue to exist thereafter, and content can no longer be migrated between them unless permitted by USI.

It is advisable to move all content from a device with a CPCM Storage Entity prior to migrating the host device, as some implementations may not permit the content to be removed once the domain migration has taken place. In the event that a problem occurs, it should be possible to move the storage device back to the original domain for the purposes of removing such content, before returning it to the new domain.

In summary, the AD Conversion Device is used to create the new AD, to migrate one or more devices to the new AD, and to migrate that content which is chosen and permitted to be migrated into the new AD.

8.7.6 Migrating Content between Authorised Domains

The following considerations are necessary when allowing the migration of Content Licenses (and hence content) between ADs in this way.

8.7.6.1 MCPCM and MLocal Content

Content that is marked as MCPCM or MLocal can be migrated directly between devices belonging to different domains, without the need for an AD Conversion Device. Normal CPCM content handling will support this.

8.7.6.2 Copy-Never Content

CN Content should never be moved between domains. CN content is Acquired for direct playback only, possibly with buffering, and there should never be a copy available to be moved. If CN content is stored on a CPCM-enabled device but is outside the CPCM system, such as still being under CA or DRM control, it is out of scope for the present document.

8.7.6.3 Copy-Once and Copy-No-More Content

For Copy-Once and Copy-No-More Content, while there may be multiple copies of the encrypted audio-visual content, there will only be a single Content License in existence at a given time. Therefore, the migration of this Content License inherently makes any media copies unusable, so they can be safely ignored and/or deleted once the migration is completed.

8.7.6.4 Copy-Control-Not-Asserted Content

For CCNA content, it is never easy to determine how many copies of a Content License exist. It is therefore usually not permitted to migrate such content between domains, unless marked as MCPCM or MLocal, as there is no certain way to ensure that all copies are migrated to the new domain.

This is not a problem when Combining ADs because it is possible, through the Domain Internal Record, to know that all CPCM Instances have left the original AD, after which it can be deleted. Because the AD Conversion Device can be sure that no ADSE-countable devices remain that hold the original AD secret, there is no harm in leaving some un-migrated Content Licenses in existence, as they will no longer be usable by any device.

If there are some other proprietary means outside CPCM to ensure that all copies of a Content License bound to the original AD have been deleted, the C&R Regime may choose to permit migration. However such an approach is out of scope for these guidelines. The same applies if there are proprietary means to ensure that all members of an AD have left or been rendered non-functional. Notwithstanding the above, if both the source and destination ADs will both continue to exist after the process is completed, and there is no other means to ensure that copies are correctly deleted, migration of CCNA content should not be permitted without clear approval of the C&R Regime.

8.7.6.5 Read-Only Storage Media

CPCM allows content to be stored on read-only media, such as burning to an optical disk. It may not be possible to rewrite such media to change the AD, which makes it difficult to ensure that Content is truly migrated and does not remain in the original AD. The only practical solution to this is that the content is migrated, then the original media immediately destroyed, recycled or otherwise made useless under suitable supervision approved by the C&R Regime.

8.8 Use of ADM by non-CPCM Content Protection Systems

The CPCM ADM subsystem can be implemented for content protection systems other than DVB-CPCM. This approach also allows the logical AD to be extended to other technologies.

8.8.1 Adoption of CPCM ADM Technology

A content protection technology may choose to incorporate elements of the CPCM ADM specification rather than design a new domain management solution. In this case, the new ADM is totally separate from CPCM and will probably not interwork with CPCM ADM.

8.8.2 Non-CPCM Devices Joining a CPCM AD

It is possible to use CPCM ADM to facilitate interoperability between CPCM and another domain-aware content protection technology.

There are several approaches available for this:

1) The non-CPCM Device may implement the CPCM ADM stack, ADSE tools, and the necessary tools from the Security Control functions. It will probably also need a CPCM certificate obtained from an appropriate C&R regime, In this case, the non-CPCM Device looks just like a CPCM Instance for the purposes of Joining or Leaving the AD.

2) The non-CPCM Device may implement a proprietary mechanism for joining the AD, perhaps using protocols defined by another content protection system. In this case, the relevant AD information such as ADID, AD Name, AD Secret, are communicated by non-CPCM protocols, subject to rules agreed with the C&R Regime, but using the same ADSE approach. From a CPCM logical perspective, this means using an AAA function at the CPCM device.

3) If a device implements a non-CPCM content protection mechanism which also includes a Domain capability, it may be possible to associate the CPCM ADID with the non-CPCM domain such that both are treated logically as a single domain. This assumes either that the non-CPCM domain is able to adequately ensure the integrity of the overall domain size and extent, or that the C&R Regime allows the two domains to share content though they are separately managed.

4) It may be possible to use an overarching "super domain" to regulate the use of domain management in both CPCM and non-CPCM protection systems. This approach requires the super-domain to have sight and control of current ADSE counts. In this case, the CPCM ADID becomes a property of the super-domain. In practice, such a super-domain might just be the domain management of another technology, to which the CPCM AD is made a subservient function.

8.8.3 Bridging between non-CPCM content protection technologies

The same approaches proposed in clause 8.8.2 can also be used to bridge between two or more different non-CPCM technologies. Essentially, the CPCM ADM approach provides a common means to identify and to control the size and extent of the overall logical domain, or of a subset of that domain. The CPCM ADID becomes a common identifier for this logical domain, which can be reused in other technologies as a common point of reference. The CPCM ADSE mechanism can be used to police the logical domain, or a subset thereof. This use of common domain identification does not automatically allow content to flow, however it may be used in combination with other mechanisms to make this easier.

8.8.4 Content Delivery to a non-CPCM Device

In many content management scenarios, a requirement is that content can only be passed to devices that are members of the same AD.

When there is a mix of content protection technologies, this becomes something of a challenge, as there may not be a common means of identifying which devices are logically associated as an AD.

The CPCM ADID can be used for this purpose.

We assume that the ADID becomes associated with the content item in some way, either directly in the content or in some kind of Content Licence. This could be a CPCM Content Licence, or perhaps more likely a licence defined by the "other" technology.

EXAMPLE 1: Content is acquired by a device that contains Acquisition Point functions for both CPCM and "DRM-X". A CPCM Content Licence is created as usual, including the ADID. In parallel a DRM-X licence is created with similar rights. The same CPCM ADID is embedded in this licence using a defined extensibility field of the DRM-X licence format.

When the content is transferred to another device under the non-CPCM protection system, the receiving device may not have sufficient rights to play the content directly. However, if it finds the CPCM ADID matches with one that it knows, it may determine that playing is permitted under special rules. This can occur even when the two devices are members of different ADs under their own protection technology such as when the non-CPCM technology has a different domain concept, perhaps a "personal" domain tied to a single user, rather than the "household" that CPCM enables.

EXAMPLE 2: Anna's device implements both CPCM and DRM-X as noted in the example above, and belongs to her personal DRM-X domain. Bob's device implements only DRM-X and belongs to his personal DRM-X domain, with extensions that allow it to know a CPCM ADID and also access content that matches the ADID. The content is acquired by Anna as described above, and transferred to Bob under DRM-X control. Anna's device sees that the DRM-X domains do not match, but the CPCM ADID does match, so the transfer is enabled by the DRM-X extension.

This above approach assumes that the non-CPCM protection system is able to encrypt content in such a way that a non-member device is able to access it when permitted. However, the CPCM ADID can also be used when this is not possible.

EXAMPLE 3: Anna and Bob each have devices that implement DRM-X and are ADM-aware. Anna acquires content protected using DRM-X. However, Anna and Bob each have their own personal DRM-X domains which do not normally enable content sharing between them, though this is permitted by the granted rights. Anna and Bob have linked their own personal DRM-X domains under a single CPCM AD. The DRM-X content licence cannot be protected using Anna's DRM-X domain key, as Bob's device will not be able to access it. DRM-X includes an ADM-aware AAA function to manage the DRM-X domains and enable additional interworking scenarios. Anna's device uses the AAA function to provide a suitable DRM-X licence to Bob's device, using the ADID to associate the two devices.

EXAMPLE 4: Anna's device implements DRM-X and CPCM. Bob's device implements DRM-Y and CPCM. Each device belongs to a personal domain, but both belong to the same CPCM domain. CPCM ADSE ensures that the whole domain size is kept within bounds. Anna acquires content from a DRM-X source, which allows export to CPCM within the AD. Anna's device reformats the content and applies CPCM domain protection to it. Anna sends the content to Bob under CPCM protection, allowed by the common AD membership. The original DRM-X licence is embedded in the CPCM Content Licence. Bob's device uses CPCM to access the protected content, and converts it to DRM-Y. This conversion is able to identify the correct rights to be enabled in DRM-Y. In this example, the DRM-X licence could also be carried in other ways, such as under SAC protection. Putting it in the CPCM Content Licence ensures that it will always be available during Export from CPCM to another system. DRM-X does not need to be aware of DRM-Y. The export to DRM-Y can be based on the original DRM-X rights if DRM-Y can understand these, or only on the CPCM USI if not. In general it is best to use the rights that are closest to the original source rights, to avoid unnecessary restrictions creeping in during conversion between rights expression formats.

9 Adaptation Layers Guidelines

9.1 MPEG-2 TS Adaptation layer

9.1.1 TS header management versus key changes

Impact on the MPEG-2 TS header of the implementation of Content Scrambling Key changes are described in clause 6.4.4.1.

9.1.2 TS header management versus MDD mode

This clause clarifies the way the transport_scrambling_control bits of the TS header are set when using the MDD mode in the LSA implementation.

The Sink CPCM instance will receive the scrambled TS with the transport_scrambling_control bits indicating the scrambling and parity applied to the TS (10 or 11). Since TS 102 825-5 [i.14] does not specify these bits need to be masked, it is clear the Sink CPCM instance needs to calculate the IVE vector with the MSC containing the transport_scrambling_control bits as received. If the Sink CPCM instance has some re-scrambling to perform (e.g. key change in SE), it also modifies the transport_scrambling_control bits after descrambling.

Reciprocally, for this to work, the Source CPCM Instance sets the transport_scrambling_control bits into the TS header before preparing the IVE and scrambling.

9.1.3 Content Licence, CPCM auxiliary data and Revocation List Data carriage

Carriage of CPCM Content Licence, CPCM Auxiliary Data and CPCM Revocation List is done using the extended descriptor CP_descriptor defined in EN 300 468 [i.10]. Each of these three CPCM elements is assigned a dedicated CP_private_identifier for a given root authority i.e. each root authority will be allocated three different identifiers. This allows the rapid identification of the relevant CPCM element for a given C&R regime.

A MPEG2-TS content item may embed several instances of each CPCM element under the same or under different CP_private_identifiers.

EXAMPLE 1: Content Revocation Lists issued by C&R regimes depending upon the same Root Authority will be carried using different SRM CP_descriptors which have the same CP_private_identifier. Revocation Lists issued by a C&R regime depending upon different Root Authorities will be carried using different SRM CP_descriptors which have different CP_private_identifiers.

EXAMPLE 2: A Content item may have different Authorized Usages for different C&R regimes. Content Licences and associated CPCM Auxiliary Data corresponding to these different Authorized Usages will be carried using several Content Licence CP_descriptors. Each Content Licence CP_descriptor will carry the CP_private_identifier of the relevant Root Authority. Therefore, a content item may carry different Content Licence CP_descriptors which have the same CP_private_identifier but different C&R regime masks.

EXAMPLE 3: A commercial offer may propose to the consumer different content usages for the same price. A typical example is a first set of restricted usages during a first time-window and more relaxed usages once this time-window has elapsed. A possible implementation of this commercial offer is the creation of two Content Licences, and possibly associated Auxiliary Data, at Acquisition, each one corresponding to a different time window. The content will then carry different Content Licence CP_descriptors having the same CP_private_identifier and the same C&R regime mask.

When the Content Licence requires SAC protection, the full message embedding the Content Licence is carried in the TS. At the Content Licence location, the message header, message body and the SAC signature are carried while the CPCM Auxiliary Data are carried at their own location.

CP_descriptor is carried in the PMT.

When CP_descriptor carries a SRM but does not specify which C&R regime is applicable to the corresponding Revocation List, implementations are advised to extract the Revocation List and then inspect whether it is applicable or not. Any other action would prevent them getting the most recent RL and then they could be denied access to the Content.

When CP_descriptor carries a SRM and specifies a C&R regime, implementations extract the Revocation List that corresponds to their C&R regimes. They then look at the RL index to determine whether the list is more recent than the one they have stored, and if so, replace the old Revocation List with the most recent one.

9.1.4 CP identifier descriptor

A CP_identifier_descriptor may also be embedded in the TS. It allows for the immediate identification of which Root Authority or Authorities are allowed for the CPCM Content. A CP_identifier_descriptor may be located in the PMT, the EIT, the SDT the BAT or the NIT, depending on which content item it applies to. If the CP_identifier_descriptor is present in more than one location, the most restrictive one takes precedence. CP_identifier_descriptor allows the determination of which PMT the actual CP_descriptor is located in.

A content item may embed several CP_identifier_descriptors carrying different identifiers. If the descriptors are located at the same level, this means that the content item is available to several Root Authorities. If they are located at different levels, the above precedence rule applies for each CP_identifier_descriptor

EXAMPLE: If content is available to two C&R regimes and the CP_identifier_descriptor for the first C&R regime is located in the BAT while the CP_identifier_descriptor of the second C&R regime is located in the SDT, the second one does not take precedence over the first one as their scope is different.

9.1.5 CPCM delivery signalling descriptor

CPCM_delivery_signalling_descriptor is used to carry CPCM usage information. It is informative and is not used to determine the CPCM USI during Acquisition unless required by a CPCM C&R regime. It provides information to the user interface software in the receiver which indicates the expected usage rules that will apply to the service when entering CPCM protection. Then, the user interface software can inform the user of the content usage rules that would apply if content were acquired within CPCM. Use of this descriptor can help in avoiding the creation of false expectations with respect to the possible uses of the content.

EXAMPLE: If a user attempts to schedule in advance the recording of an event on a service marked copy never, the system can inform them that this operation cannot be successfully achieved.

When present in the SDT, the CPCM_delivery_signalling_descriptor defines a default set of expected usage rules that is inherited by all events forming part of this service.

NOTE: Any such default set of usage rules may be overridden on an event by event basis by the use of the CPCM_delivery_signalling_descriptor in the EIT.

This descriptor is allowed only once in the loop. Transmission of this descriptor is optional.

When present in the EIT, the CPCM_delivery_signalling_descriptor defines a set of usage rules that applies only to the current event. Transmission of this descriptor is optional.

This descriptor is common to CPCM C&R regimes, including when they depend upon a different root authority.

9.1.6 Content Licence and Auxiliary Data Insertion

9.1.6.1 Free-To-Air Content

9.1.6.1.1 General Behaviour

This clause is applicable to FTA broadcast content that reaches a CPCM FTA Acquisition Point. It is recommended in this case for broadcasters to insert a stuffing descriptor to reserve space in the PMT for the CL and possibly Auxiliary Data to be inserted at the Acquisition Point.

In addition, one or two new PIDs carrying the actual CL and possibly Auxiliary data will need to be added to the stream at the Acquisition Point. Unless otherwise defined by the applicable C&R, the Acquisition Point may choose any PID that is not already in use in the received TS. As there is no means to predict which PID values may appear in the stream, higher values are recommended.

Upon Acquiring content into CPCM, the stuffing descriptor length will be set to 0, reducing the total size of the stuffing descriptor to two bytes. Handling of other stuffing bytes is detailed in the following clauses.

9.1.6.1.2 Content Licence only insertion

In the case where only a Content Licence needs to be inserted, it is recommended for broadcasters to insert a stuffing descriptor of at least 9 bytes total size in the PMT.

Broadcasters are thus also recommended to ensure that NULL packets (PID = 0x1FFF) are present in the TS at a minimum rate of 3 kbit/s (i.e. two TS packets per seconds) per FTA service to be acquired. This recommendation of at least two TS packet of NULL per second is in addition to any other requirements for a minimum amount of NULL packets such as in TS 101 211 [i.21], clause 4.1.1.

At the Acquisition Point, the 7 stuffing bytes of the stuffing descriptor will be replaced with the CP_descriptor, corresponding to a CPCM Content Licence.

NULL packets which are present in the TS will be replaced with a Content Licence PID.

9.1.6.1.3 Content Licence and Auxiliary Data insertion

In the case where both a Content Licence and Auxiliary Data need to be inserted, it is recommended for broadcasters to insert a stuffing descriptor of at least 16 bytes total size in the PMT.

Broadcasters are thus also recommended to ensure that NULL packets (PID = 0x1FFF) are present in the TS at a minimum rate of 6 kbit/s (i.e. two TS packets per seconds) per FTA service to be acquired. This recommendation of at least four TS packet of NULL per second is in addition to any other requirements for a minimum amount of NULL packets such as in TS 101 211 [i.21], clause 4.1.1.

At the Acquisition Point, the first 7 stuffing bytes of the stuffing descriptor will be replaced with the CP_descriptor, corresponding to a CPCM Content Licence. The 7 following bytes will be replaced with the CP_descriptor corresponding to CPCM Auxiliary Data.

NULL packets which are present in the TS will be replaced with a Content Licence PID and Auxiliary Data PID.

9.1.6.2 CAS/DRM Content

9.1.6.2.1 General Behaviour

This clause is applicable for the broadcast of scrambled content carrying proprietary CAS or DRM ECMs. It is recommended in this case for broadcasters to insert a stuffing descriptor to reserve space in the PMT for the CL and possibly for Auxiliary Data to be inserted at the Acquisition Point.

The CA_descriptor can be replaced with a stuffing descriptor of two bytes total size, if no further access to ECMs is needed after the content Acquisition.

In addition, one or two new PIDs carrying the actual CL and possibly Auxiliary data will need to be added to the stream at the Acquisition Point. Unless otherwise defined by the applicable C&R, the Acquisition Point may choose any PID that is not already in use in the received TS. As there is no means to predict which PID values may appear in the stream, higher values are recommended.

Original CA ECMs are replaced with CPCM Content Licences and, if applicable, CPCM Auxiliary Data as appropriate. It may not be necessary to replace all of them, but a minimum number of CPCM Content Licences and, if applicable, Auxiliary Data, is recommended to be inserted to ensure that receivers provide the user with acceptable service acquisition behaviour.

9.1.6.2.2 Content Licence only insertion

In this case, it is recommended for broadcasters to insert a stuffing descriptor of at least 5 bytes total size to reserve space in the PMT. Upon Acquiring content into CPCM, the stuffing descriptor is replaced with a CP descriptor corresponding to the CPCM Content Licence

9.1.6.2.3 Content Licence and Auxiliary Data insertion

In the case where both Content Licences and Auxiliary Data need to be inserted, a stuffing descriptor of at least 12 bytes total size is recommended to be inserted in the PMT. Upon Acquiring content into CPCM, the stuffing descriptor is replaced with one CP descriptor corresponding to the CPCM Content Licence and one CP_descriptor for the CPCM Auxiliary Data.

9.1.6.2.4 Alternative behaviours

Alternate solutions to the one described above also exist. Some of them are outlined below:

1) The stuffing descriptor can be reduced to 3 bytes total size (10 bytes if Auxiliary Data need also to be inserted) and inserted adjacent to a CA descriptor. Upon Acquiring content, the whole of a CA descriptor and stuffing descriptor will be replaced with a stuffing descriptor of two bytes total size and a CP descriptor corresponding to a CPCM Content Licence (and possibly a second one for CPCM Auxiliary Data, if appropriate)

2) The CA descriptor's size may be increased to 7 bytes, adding one extra private byte that is ignored by the CAS or the DRM. This CA descriptor will then be replaced with a CP descriptor. If Auxiliary data need also to be inserted, the CA_descriptor size may be extended to 14 bytes, and the CA descriptor is replaced with two CP_descriptors.

9.2 File Format Adaptation Layer

9.2.1 General

DVB-FF (TS 102 833 [i.15]) specifies the File Format used to store and playback DVB services. It is built on top of ISO base media file format [i.22].

ISO base media file format allows separating content from other information, including details of used codecs, locations of samples for codecs, and indices for random access points. The specification defines tracks and hint tracks. A track is a collection of related samples. A hint track is a special track which does not contain media data but does contain instructions for packaging one or more tracks into a streaming channel. DVB-FF extends the ISO base media file format to allow the storage of audio, video and other content as reception hint tracks. Reception hint tracks are used when one or more packet streams of data are recorded. Reception hint tracks indicate the order, reception timing, and contents of the received packets.

Formally, ISO base media file format files consist of a sequence of imbricate boxes, which are building blocks specified by their size and a Four Character Code (4CC). Each box may include a number of attribute fields and other boxes as sub-boxes. Basically, the `'moov'` box is a container for all the metadata. It encapsulates a track box for each individual track or stream; the media information in a track includes sample description boxes `'stsd'` which contain sequences of sample entries detailing codecs used in that track; the following clauses refer mainly to the `'stsd'` box.

Thus, DVB-FF extends ISO base media file format by specifying:

- Sample entries for new track types, in particular Protected MPEG-2 TS reception hint track 'pm2t'.
- Extension to a number of existing boxes of the ISO base media file format, in particular the protection scheme information `'sinf'` box, and the content license `'clic'`, and auxiliary data `'caux'` sub-boxes.
- New boxes, in particular the System Renewability Message container box `'srmc'` and the System Renewability Message box `'srmb'`.

DVB-FF defines a general mechanism to store protected MPEG2-TS and RTP streams. It works by changing the 4CC of the sample entry of the reception hint tracks, and appending boxes containing the original 4CC followed by details of the protection mechanism and related data. DVB-FF applies this mechanism to signal CPCM protection of MPEG-2 TS reception hint tracks by specifying boxes for license, auxiliary data, and revocation list objects and defining values of attributes to indicate CPCM types.

The following clauses explain how CPCM is adapted to DVB-FF by detailing which DVB-FF boxes the CPCM elements and related information are mapped and what field values need to be used to correctly indicate CPCM usages.

This adaptation is independent of the CPCM adaptation to the specific content format.

Normative adaptation layers are given in TS 102 825-9 [i.11] and TS 102 833 [i.15].

9.2.2 Common CPCM adaptation

This clause explains the mapping of CPCM Content to DVB-FF elements that is independent from the content format. The following applies to all adaptations layers:

- Files have the extension `'.dvb'`.
- Each such file has the file type box `'ftyp'` at the beginning; further setting of the fields in this box depends on the content format.
- Revocation lists are stored in the System Renewability Message container box `'srmc'` containing as many System Renewability Message sub-boxes `'srmb' as there are messages to carry`. The `'srmb'` box signals the CPCM CP system ID.

9.2.3 CPCM adaptation to DVB-FF for MPEG2-TS

This clause explains the mapping of CPCM content to DVB-FF storage of MPEG-2 TS. It is assumed that the CPCM Content to be stored is in a format that conforms to the MPEG2-TS specification and to the CPCM adaptation layer for MPEG2-TS.

The adaptation of CPCM content to a DVB-FF file uses the following elements:

- The file type box `'ftyp'` has:
  - the `major_brand` field set to `'dvt1'` (i.e. DVB Transport Stream brand); and
  - the `minor_version` field set to = `x*256 + y` for `x.y.z` version of the brand of TS 102 833 ([i.15])

EXAMPLE: Using TS 102 833 [i.15] v1.1.1, the brand version is set to 257.

- Each sample entry of enabled tracks is set to `'pm2t'`, meaning Protected MPEG 2 Transport Stream reception hint track.
- The `'pm2t'` sample entry includes a protection scheme information box `'sinf'`.

NOTE: `'rm2t'` cannot be used instead of `'pm2t'` even in case of DNCS-marked CPCM Content, since the `'sinf'` box is a sub-box of `'pm2t'` and is needed to carry the license.

- The `'sinf'` box includes the original format box `'frma'` with the `data_format` field set to the 4CC `'rm2t'`, indicating the format of the decrypted, encoded data, i.e. the original MPEG-2 TS reception hint track sample entry as specified for unprotected MPEG-2 TS.
- The `'sinf'` box includes the scheme type box `'schm'` describing the scheme type with:
  - `scheme_type` field set to `'cpcm'`; and
  - `scheme_version` field set to `1` (version of CPCM encapsulation, not of CPCM).

- If the Content Licence and the Auxiliary Data carriage is out of band, the 'sinf' box includes the scheme information box 'schi' with:
  - a sub-box 'clic' for a CPCM Content License; and
  - a sub-box 'caux' for CPCM Auxiliary Data.
- Multiple 'pm2t' sub-boxes of 'stsd' boxes are used if multiple licenses and auxiliary data have to be stored.

These mappings are summarized in table 8.

Table 8: DVB-FF boxes and fields setup for CPCM adaptation

Boxes										Fields	Comment
ftyp										major_brand='dvt1'	Beginning of file
moov											
	track										
		mdia									
			minf								
				stbl							
					stsd						
						pm2t					One per CL / Auxiliary Data
							sinf				Protection scheme information
								frma		data_format='rm2t'	Original format
								schm		scheme_type='cpcm' scheme_version=1	Scheme type
								schi			Scheme information
									clic		CL
									caux		Auxiliary Data
srmc											SRM container
	srmb									CP_system_ID	SRM (RL for CPCM CP system)
NOTE: The first six columns indicate the level of the box; only impacted boxes are indicated, other are omitted											

9.3 Home Network Ecosystems Adaptation Layer

DVB-CPCM has been designed to be network agnostic. Thus, dependence upon specific properties of the HN layer has been reduced to the minimum, allowing CPCM to be kept largely independent from the underlying HN technology. Only elements that are external to CPCM or to CPCM interoperability framework need to be managed by the HN layer, namely:

- The HN layer provides device and content discovery capability. This includes non-CPCM devices and non-CPCM content.
- The HN layer provides messaging capabilities: This includes point-to-point messaging and broadcasting as defined in TS 102 825-4 [i.9].
- The HN layer communicates the version of CPCM that is supported by each CPCM Instance.
- For Acquisition Points, the HN layer has to advertise from which external CP systems content can be Acquired.
- For Export Points, the HN layer has to advertise which external CPS content can be Exported to.
- HN layer has also to convey cp_system_id, which is included in cp_descriptor, defined in clause 9.1.3, and is used to enable interoperability between two CPCM C&R regimes depending upon different Root Authorities.

The HN-Layer acts as an intermediary between two CPCM Instances embedded on two different devices. The CPCM layer sends CPCM messages via the HN layer. The HN layer forwards the message across the network and to the relevant CPCM instance in the destination device. The HN layer in the sink device identifies which CPCM Instance is the addressee and forwards the message to the relevant CPCM layer, together with the identity of the source CPCM Instance.

Management of communications or discovery between two CPCM Instances embedded on the same device are implementer's choice, but still subject to approval of relevant CPCM C&R regimes.

The HN layer also provides transmission of CPCM Revocation Lists.

9.3.1 DVB-HN Adaptation Layer

9.3.1.1 Introduction

The DVB-HN (TS 102 905 [i.23]) networking layer is based on UPnP. Each CPCM Instance is represented as a distinct UPnP service. It allows an external CPCM Instance to address separately CPCM Instances that would be embedded in a same CPCM device.

9.3.1.2 CPCM Instance Discovery

DVB HN layer is in charge of the discovery of all devices that are reachable through the network.

For CPCM devices, relevant information is provided as follows:

- CPCM version is provided in the UPnP service type description.
- CP_system_id is the prefix of the service id name.
- CIC identifier is the suffix of the service id name.
- The list of external systems from which content can be Acquired or to which content can be Exported is obtained using GetCapabilities() Action.

Extra information on CPCM Instance capabilities or status can be obtained using one of the following CPCM messages:

- CPCM status enquiry which allows for collecting static information extracted from the certificate, and dynamic information (i.e. AD related information) of a particular CPCM Instance.
- CPCM discovery request enquiry which allows for collecting static information, which is extracted from the certificate, and dynamic information such as AD related information, of all present CPCM Instances.
- CPCM AKE initiation which starts Trust establishment with a given CPCM Instance and allows collection of certificate information.

Choice between above options is left to implementers and depends from the intended action. For instance, if AD management is expected, CPCM discovery request is preferred as it is the first step of AD management.

The discovery of non-CPCM devices is important in the following cases:

- Discovery of media servers where CPCM content could be stored (bit bucket scenarios).
- Discovery of external CPS devices with which content might be exchanged, through CPCM AP or EP.

9.3.1.3 Content Discovery

DVB HN layer is in charge of the discovery of all content that are available in the network. CPCM content requires a dedicated XML fragment. The following elements are directly available at the HN layer:

- CPCM version is provided in the XML fragment.
- CP_system_id is the prefix of 'CPCMInstanceId' element.
- CLID of the associated content licence. If a content item is associated with several Content Licences having different CLIDs, it is considered to be different CPCM Content Items and thus will be exposed as one Content Item per CLID.

- The content licence protection mode (SAC/instance secret protected and/or AD Secret protected). This allows the sink CPCM Instance to know whether it needs to establish a SAC with the source before trying to obtain the Content Licence.

- The content licence transport mode (in-band or out-of-band) which allows implementations to determine whether the CPCM Instance needs first to obtain CPCM Content Licence in a dedicated message, or if it can only request the content with an embedded Content Licence.

Other CPCM information on content (mostly authorized usage) can be obtained using CPCM messages:

- CPCM content discovery request message to get authorized usage and AD status of the content.

- CPCM get CL message to get the full Content Licence directly. This is possible only if the content licence is available out-of-band. If the Content Licence requires SAC protection, the implementation will first establish a SAC with the source CPCM Instance, if none was previously established.

Choice between above options is left to implementers and depends on the intended action. For instance, if authorized usage is only needed to display the relevant information to the user, CPCM Content Discovery protocol is preferable as it is lighter.

9.3.2 Proximity Tools Adaptation Layer

9.3.2.1 Introduction

Four additional tools are proposed for the DVB-HN layer. Three of them make usage of the availability of TTL ([i.24]) on IP networks. The last one heavily exploits characteristics of IP networks.

The specification suggests that RTT and SRTTL mandatory tools may be respectively replaced with RTT and TTL and respectively SRTTL. As already explained in clause 6.3.5.5.1, a C&R regime has to be cautious before taking such a decision as there is no guarantee that the IP link will be used on the entire communication path. Hence, TTL values might be lost and the test may fail when the two devices are actually in proximity.

9.3.2.2 RTT

On an IP network, RTT uses a 'ping' message that might be blocked by firewalls and the tool might not work. While in the past it might have been considered that firewalls would only appear at the boundary of the home network, this assumption is no longer true. Today, more and more devices implement a "personal firewall" that might also block ping requests, even when it comes from within the Local Environment. So, it may be that no response is received even from a Local device. Thus, unless otherwise required by the applicable C&R regime, the RTT test may be skipped since CPCM has other tools to ensure that content is not redistributed remotely (see clause 6.3.5.5.1).

EXAMPLE: When proximity-restricted Content is stored on a bit bucket and then passed to a Consumption Point, the Consumption Point will analyse the Content Licence and determine that the Content is proximity restricted and which Storage Entity stored it on the bit bucket. It will then perform a proximity test with that Storage Entity to determine whether the Content is still in the Local Environment. Thus, even if RTT tests could not be performed between the Storage Entity and the bit bucket and/or the Consumption Point and the bit bucket, content consumption may only happen in the Local Environment.

9.3.2.3 TTL

TTL is the standard mechanism of IP (RFC 791 [i.24]). The device sets the ttl_value of the message it is sending to the ceiling value TTL_value. The idea is that if the message has to make more hops than TTL_value to get to the sink device, it will get lost. Thus, implementations using TTL will not get a ping response (but an error message 'ICMP time exceeded') and may determine, possible after several trials, that the CPCM Instance is remote. The test may be done with the actual message that needs to be sent, or with a 'ping', or with any other message, CPCM or not.

The TTL test is not fully accurate for at least the following reasons:

1) Ethernet hubs, switches, and some routers will fail to decrement the TTL value when they route a message. Hence, some network configurations may allow TTL-limited messages to travel beyond the Local Environment.

2) The ttl_value field of the message is not secured and thus can be altered. An attacker could thus systematically increase all TTL values of outgoing messages of CPCM devices to defeat this test.

Thus, even if the TTL proximity method fails, if authorized by the applicable C&R regime, a CPCM Instance may decide to perform anyhow as CPCM has other tools to control that content was not redistributed remotely (see clause 6.3.5.5.1).

EXAMPLE: A CPCM Storage Entity may decide to proceed to the recording of a Content Item on a bit bucket even though all TTL proximity tests previously failed.

9.3.2.4 STTL

Secure TTL is a TTL mechanism secured against attacks based on increasing the ttl_value field. It may be used only between two CPCM Instances. The protocol works by exchanging messages whose ttl_value fields are randomly set, but to a value which is higher than the authorized ceiling, and, upon receiving the message, comparing the received ttl_value field with the one set at sending and determining whether the TTL ceiling was exceeded or not.

In the STTL protocol, the request message is empty but the response message includes the final value of the TTL of the request message and the original value of the TTL of the response message. The response message is authenticated. Upon receiving the response message, the requesting CPCM Instance makes a double check:

- It compares the original TTL value of the request message with the final one received in the message. If the number of hops is greater than the ceiling or if it is negative, it issues an error code "STTL request message out of range: proximity test failure".
- It compares the original TTL value of the response message included in the message with the final one that he measured. If the number of hops is greater than the ceiling or if it is negative, it issues an error code "STTL response message out of range: proximity test failure".

If no error message is generated, the test is successful. Note that there is no preference on the order in which verifications should be done. Thus, when both TTL values are out of range, different implementations may issue different error codes.

NOTE: STTL does nothing about Ethernet hubs, switches or routers that do not decrement the ttl_value field. Also, it can still be defeated by careful manipulation of the ttl_value field. However, such manipulation requires a far more complex hack than for the regular TTL mechanism.

9.3.2.5 SRTTL

SRTTL is a straightforward combination of SRTT and TTL. Instead of exchanging dedicated message protocols to measure the number of hops, TTL values are set upon the two initial messages of SRTT. In case of RTT success, received TTL value of the request message and original TTL value of the response message are inserted in the last protocol message of SRTT protocol, which is already authenticated. Hence, the combination requires exactly the same number of messages as SRTT.

Specific error codes for SRTTL are the combination of the ones for SRTT and TTL.

9.3.2.6 NTT

TS 102 825-9 [i.11] suggests concepts for NTT tools rather than specifying any of these tools. The underlying idea is to send special IP packets and to determine whether they are received or not. These test packets should be such that they only produce false positive or false negative.

EXAMPLE: If Local devices always receive the test packet but Remote devices may receive it or not, the test will only produce false positives since some Remote devices may receive the packet.

This test may then be used in combination with other tests (NTT or not) to ascertain the result.

Implementers have to get their test endorsed by the governing C&R regime before they can use it. This kind of test is very flexible since it needs only to be implemented on the testing device and thus does not cause any interoperability issues.

NOTE: A C&R regime may also select and approve a NTT tool requiring cooperation between different CPCM implementations (e.g. using extension messages, or using a particular platform capability) to create a federated picture of the local network topology. Such an approach is out of scope for the CPCM specification.

9.4 Physical Interfaces Adaptation Layers

9.4.1 CI Adaptation Layer

9.4.1.1 Introduction

This clause provides guidelines for implementing a CPCM layer on top of the current DVB Common Interface for the purpose of securing the link between DVB Common Interface Modules ([i.25] and [i.26]) and Set-Top-Boxes or Integrated Digital TV.

9.4.1.2 Acquisition of content from CA systems

9.4.1.2.1 System Overview

A Common Interface Module receives typically CA protected content from a CI Host device such as an iDTV or a Set Top Box. The Host device usually includes tuners to receive the broadcast stream as well as the video decoder. The content is passed between the Host and CICAM as an MPEG2 Transport Stream. Therefore the MPEG-2 TS adaptation layer as specified in TS 102 825-9 [i.11] should be used.

The full adaptation layer of CPCM to DVB Common Interface is described in TS 102 825-9 [i.11].

In particular the method of identification of CPCM copy protection resources and the mapping of the CPCM Protocol messages to the CI Command Interface are all defined. The Copy Protection ID allocated by IEEE to CPCM is the DVB one namely 0x000410C1.

The figure 22 represents the system overview using CPCM to secure the link between DVB Common Interface Modules and Host in Set-Top-Boxes or Integrated Digital TV.

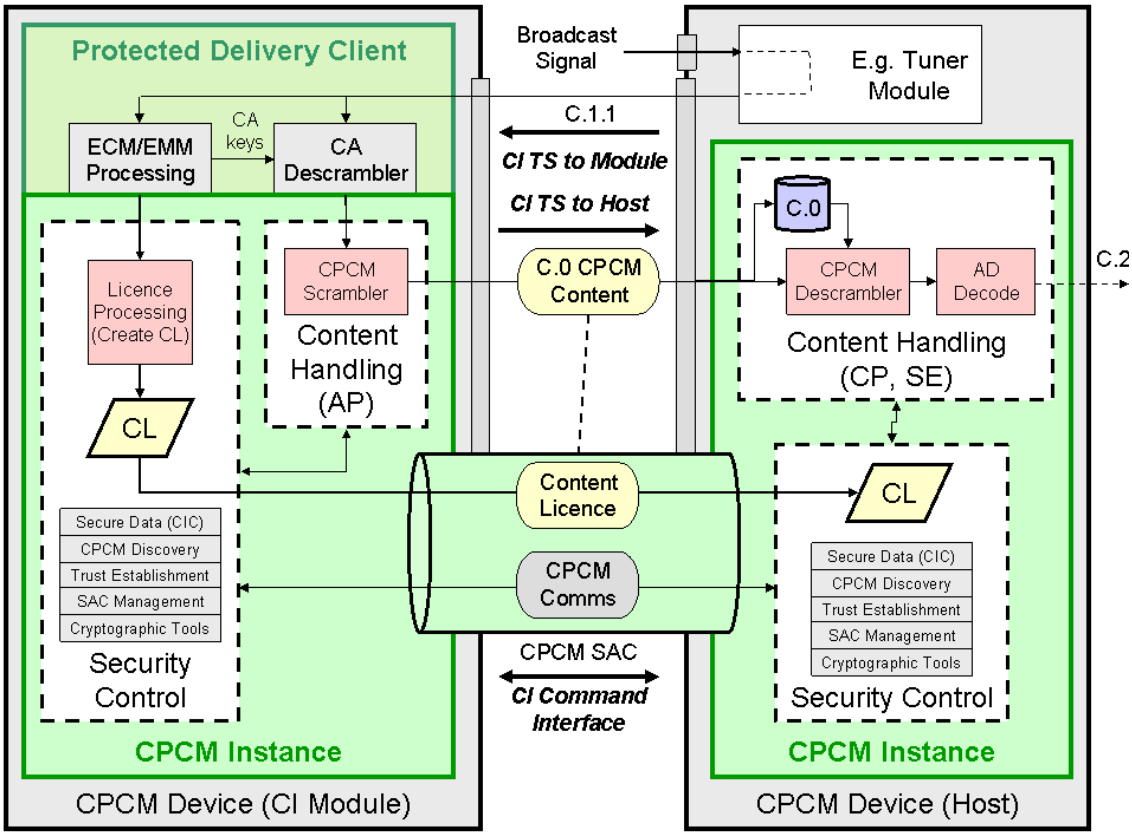


Figure 22: Minimal subset for Acquisition of protected input content into CPCM via a CI Module

Both Device and Host are mutually authenticated using the CPCM standard mechanism. CPCM does not provide nor need any pairing mechanism to ensure a one-to-one matching between a CI module and its host as Authorized Domain Management and Proximity enforcement provides the expected guarantees.

In the CI Module the following takes place:

- Mutual Authentication with the Host takes place and SAC is established.
- The Content is descrambled by the Conditional Access.
- The Content is then Acquired into CPCM (Encryption using LSA and Creation of the Content Licence).
- The CPCM protected Content is sent to the CI Host in a normal Common Interface operating mode.
- The Content Licence is sent to the CI Host via the SAC.

In the CI Host the following takes place:

- Mutual Authentication with the CI Module takes place and SAC is established.
- CA protected content is sent to the CI Module.
- CPCM protected content is received from the CI Module.
- Content Licence is received via the SAC.

9.4.1.2.2 CPCM security Toolbox

Table 9 details which CPCM elements are to be implemented with regard to the CI element.

Table 9: CI elements implementation vs. CPCM functionalities

CPCM element	CI Host	CI Module
SAC	Yes	Yes
Certificate verification	Yes	Yes
Revocation management	Yes	Yes
CPCM scrambler	No	Yes
CPCM descrambler	Yes	No
Content decoder	Yes	No
Proximity Tools	Yes	Yes
ADM	Recommended	Recommended
Content Licence issuing	No	Yes
Content Licence verification	Yes	No
Secure Relative Time	Yes	Yes
Secure Absolute Time	Optional	Optional
Geographical awareness	Optional	Optional

9.4.1.3 Example Usage scenarios

Usage scenarios are defined by the type of functionalities present in Host and Modules.

9.4.1.3.1 Module

Table 10: Example usage scenarios for CI module

CPCM Functionality in CI Module	Usage Scenario
None	The Module can behave as a "pipe" for CPCM content to the Host (e.g. CPCM content located in the home network).
Acquisition	The module can acquire content into CPCM for example from a CA controlled content.
Storage	The module can control the CPCM storage functionality independently of where the content is stored (Host, Module, Network, Removable Media...). The Content Licence associated with the CPCM content can be created in such a way that the Conditional Access keeps control over the content i.e. CA controlled PVR.
Export	The Module can export the protected content to other Content Protection System (e.g. DTCP-IP).
Processing	For example the Module could embed a watermark with the content or downscale resolution...
Consumption	For example the Module could include a HDMI output.

9.4.1.3.2 Host

Table 11: Example usage scenarios for CI host

CPCM Functionality in CI Host	Usage Scenario
None	Limited usage scenarios. The Host can only send content to the CI Module but it cannot receive CPCM protected content back.
Consumption	The Host can consume CPCM protected content (e.g. iDTVs).
Storage	The Host can control the CPCM storage functionality independently of where the content is stored (Host, Network, Removable Media...).
Export	The Module can export the protected content to other Content Protection System (e.g. DTCP-IP).
Processing	For example the Module could embed a watermark with the content or downscale resolution...
Acquisition	Limited usage scenarios related to the Common Interface.

9.4.2 ISO7816 Adaptation Layer

9.4.2.1 Introduction

The ISO7816 Adaptation Layer enables the implementation of a CPCM Instance (or part of it) on a smart card by specifying the following:

- Sending and receiving CPCM messages over the smart card interface.
- Performing a proximity test between a CPCM Instance hosted on a smart card and any other one.

The use of the adaptation layer is optional when the smart card and its host both embed modules related to a proprietary CAS, DRM or CPS. In this case, the CAS, DRM or CPS may define its own adaptation layer. The adaptation layer described in TS 102 825-9 [i.11] is mandatory in all other cases.

The adaptation layer is valid for both contact and contactless interfaces.

9.4.2.2 Messaging

Defined messaging is generic and allows exchanging any CPCM message. It is future proof for any new CPCM message that may be introduced in a future version of this multi-part deliverable or by a CPCM or proprietary extension.

The idea is that only two commands have been defined:

- Send_message that is a generic command allowing for the sending of any CPCM message to the smart card.
- Get_message that is a generic command allowing for the reading of any CPCM message prepared by the smart card.

9.4.2.2.1 send_message

The send_message command body embeds two parameters:

- The CIC identifier of the CPCM Instance that issued the message, if parameter P1 is 0. If it is 1, the message comes from the host application and this parameter is absent.
- The CPCM message itself.

If the size of the two above parameters is greater than MAX_IN, typically 255, the message has to be cut into several blocks. Parameter P2 is then used to signal how many blocks of the message are yet to be sent to the card so that the card knows when it has received the entire CPCM message. The cutting into blocks of the message can be done by block of MAX_IN bytes to optimize the number of commands that need to be used or, alternatively by ensuring that no message parameter is split over two blocks.

A Status Word that may be generated by the smart card can be the following:

- 0x9000 if no error was found in the message and either the message is not yet fully transmitted or it is fully transmitted and the smart card has not prepared any response.
- 0x9AXX if the message was fully transmitted, no error occurred and the smart card has prepared a response from which XX bytes can be read immediately.
- 0x6FB1 if the host tries to send a new message while the smart card has prepared a response that was not fully read.
- 0x6FB2 if the sent CPCM message is unknown to the smart card. When the message is sent in several blocks, it is recommended that the smart card signals this error as soon as it can and does not wait until all the message blocks are received.

9.4.2.2.2 get_message

The get_message command body embeds two parameters:

- The CIC identifier of the CPCM Instance for which the message is destined, if the message is destined for a CPCM Instance. This is the case if parameter P1 is 0. If parameter P1 is 1, the message is for the host application and this parameter is absent.
- The CPCM message itself.

If the size of the two above parameters is greater than MAX_OUT, typically 256, the message has to be cut into several blocks. The smart card in that case will signal, after each block retrieval, how many bytes are ready to be read.

A Status Word that may be generated by the smart card can be the following:

- 0x9000 if no error was found in the message and the prepared message was fully read.
- 0x9BYY if YY bytes are yet to be read from the prepared response.

NOTE: If, after having being informed by the smart card that YY bytes have been prepared, the host tries to read XX bytes where XX is lower than YY, it is recommended that the XX bytes are read and the status words are set to 0x9BYY with an updated value for YY. If XX is greater than YY, 0 bytes will be read and the status words will be set to 0x9BYY with an unchanged value for YY.

- 0x9AXX if the message was fully read, no error occurred and the smart card has prepared another response from which XX bytes can be read immediately.
- 0x6FB0 if the command get_message was sent while no message has been prepared.

9.4.2.3 Proximity Control

9.4.2.3.1 Introduction

The ISO7816 smart card interface may be very slow and is thus not suited for the standard SRTT proximity method. TS 102 825-9 [i.11] proposes a tool dedicated to that interface. C&R regimes may decide to adopt this tool or to define another one.

The proposed tool builds on three different tools:

- PTDC between the smart card and its host.
- SRTT between the host application and any other CPCM Instance or host.
- PTA to allow assessing proximity between a smart card and any other Instance from the two above tools.

Implementation guidelines for SRTT and PTA may be found in clauses 6.3.5.5.3 and 6.3.5.5.5 respectively.

9.4.2.3.2 Usage scenarios

The different usage scenarios are as follows:

1) The smart card wants to assess proximity with its host: in this case it runs PTDC with its host directly.
2) The host wants to assess proximity with its smart card: to that end, it sends a PTA request to the smart card including its own CIC identifier and no proximity method. The smart card knows the request comes from its host, as P1 is 1, uses PTDC and checks that the received identifier corresponds to the one of its host.
3) The smart card wants to assess proximity with a distant CPCM Instance which is not located on a smart card. In this case, it first assesses proximity with its host, using PTDC, then it uses PTA tool, requesting its host to assess proximity with the distant CPCM Instance by inserting the corresponding CIC identifier in the PTA request and PTDC as the proximity tool identifier.

4) A distant CPCM Instance which is not located on a smart card wants to assess proximity with a smart card: In this case, the distant CPCM host requests a SRTT test that will reply with error 'SRTT not supported, use PTA'. If it supports PTA, it then send a PTA request to the smart card carrying its own identifier and no proximity method, the smart card performs a PTDC test with its host and sends a PTA request carrying PTDC as a proximity method to its host to perform a proximity test with the distant CPCM Instance.

5) A smart card wants to assess proximity with a smart card plugged onto the same host: the smart card performs a PTDC test with its host and then sends a PTA request to its host carrying PTDC as the proximity method identifier and the CIC identifier of the card to test. The host will know that it corresponds to one of the smart cards plugged into it and will behave as 2) above.

6) A smart card wants to assess proximity with a smart card plugged onto a distant host: the smart card performs a PTDC test with its host and then sends a PTA request to its host carrying PTDC as the proximity method identifier and the CIC identifier of the host to test. The host will know that it corresponds to a distant CPCM Instance and will behave as 4) above.

9.4.2.3.3 PTDC implementation guidelines

Guidelines are similar to the ones of SRTT (see clause 6.3.5.5.3).

As SRTT messages are directed to the host application, parameter P1 is set to 1 for all messages and the CIC identifier of the CPCM Instance hosted in the application does not need to be included.

9.5 Adaptation Layer for delivery networks

9.5.1 DVB delivery Adaptation Layer

9.5.1.1 CLID Allocation schemes

9.5.1.1.1 DVB Broadcast Event and Acquisition Device

This scheme uses DVB event triplet (network id, service id, event id) and the CIC identifier. One byte can be freely allocated in the case where the same CPCM Instance acquires, processes or stores the same DVB event several times.

9.5.1.1.2 DVB broadcast service and Acquisition Device

This scheme uses the DVB broadcast network id and service id and the CIC Identifier of the Acquiring CPCM Instance. Three bytes (about 16 million) can be allocated in the case where the same CPCM Instance acquires, processes or stores content from the same DVB service several times.

9.5.1.1.3 CA system and Acquisition Device

This scheme uses CA identifier of the CA from which content is Acquired and the CIC Identifier of the Acquiring CPCM Instance. Five bytes (about 10^9) can be allocated in the case where the same CPCM Instance acquires, processes or stores content from the same CAS several times.

9.5.1.1.4 DVB-MHP application identifier and Acquisition Device

This scheme uses MHP application identifier of the MHP application running in the CPCM Device hosting the Acquisition Point and the CIC identifier. This scheme may thus only be used 256 times by the same combination of Acquisition Point, Processing Entity or Storage Entity and MHP application, which makes this scheme unlikely to be used.

9.5.2 IP delivery Adaptation Layer

9.5.2.1 SRM delivery

CPCM Revocation Lists may be delivered using the SRM delivery mechanism described in TS 102 034 [i.27]. In such case, the SD&S service advertising the CPCM Revocation List also advertises the applicable C&R regime and uses a local 1-byte index allowing implementations to determine whether a newer Revocation List is available or not.

Implementations may ignore Revocation Lists that are not applicable to them.

It is advisable to download a Revocation List with matching C&R regime mask as soon as the latter changes. Since the SD&S service uses only one byte to signal the index, a lower index may correspond to a newer list. After downloading the Revocation List, implementations need to check the received Revocation List is actually newer than their stored one and store it if so.

10 Mappings Guidelines

10.1 Conditional Access Mapping

10.1.1 Scope

This clause applies to Conditional Access Systems (CAS) that are being used to deliver content that can be handed over to DVB-CPCM control. The CAS has to provide any information needed for proper mapping. The CAS is supposed to be a Trusted Source for CPCM.

The goal is to bring out the CAS tasks to be implemented in the process of CPCM-USI mapping, and clarify its responsibility. It tackles CPCM-USI insertion at CAS Head-End (HE) and extraction at AP.

From CAS implementation point of view, the mapping task can be split in:

- Management of Usage Rules at Head-End side (acquisition, preparation, content binding, signalling, secure carriage).
- Mapping of received information to CPCM USI at Acquisition.
- Possibly, inverse mapping at Export.

10.1.2 Transparent carriage and rendering of CPCM-USI

According to TS 102 825-3 [i.7], the CAS CPCM-content delivery is recognised as a Trusted Source for CPCM, i.e. there is an approved mapping between the CAS Usage Rules and CPCM USI. A simple mapping case is proposed in this clause.

It is supposed the CPCM USI related to a given content is known by the HE. For instance, it is the task of an external system to deliver to the HE the content with the CPCM USI. The handling of CPCM-USI before acquisition by the HE is under operator and content provider control.

In this case, the CAS implementation comes down to delivering and accurate rendering of this pre-defined CPCM USI at the Acquisition Point. CAS implementation then handles:

- At HE:
  - Receiving the CPCM-USI with the content.
  - Signalling the CPCM-USI in the appropriate stream.
- At the AP:
  - Creation of the CL including the CPCM-USI.

The HE can handle both Authorised Usage signalling, as defined in TS 102 825-3 [i.7], and informative signalling, as discussed in TS 102 825-9 [i.11]). The Authorised Usage signalling will result in the internal CPCM USI (i.e. the USI set in the CL at AP). The informative signalling is aimed at being used by user interface in the Home Network.

Hence, at the HE, the CPCM USI is inserted as:

- Authorised Usage signalling. This can be done either by:
  - Using any secure mean under the CAS control. The security level should match the one applied in theCPCM system for the CL protection and binding to the CPCM content.
  - Or relying on the informative signalling, but this is allowed only if required by the C&R regime.
  - Informative signalling. This can be done either by:
    - Using a cpcm_v1_delivery_signalling data structure as specified in TS 102 825-4 [i.9]. This data constitutes the body of the `cpcm_delivery_signalling_descriptor`, which can be inserted in clear in DVB-SI.
    - Or relying on the Authorised Usage signalling.
    - Or using a proprietary private descriptor.

At the AP, the extraction of the CPCM USI may be split into:

- Management of the Authorised Usage signalling. This consists of mapping the Usage Rules constituting the Authorised Usage to internal CPCM-USI, which are inserted in the CL. This is clearly under CAS control if the signal is secured by the CAS HE.
- Management of the informative signalling, which is actually not part of the CAS implementation, unless it is extracted from the Authorised Usage signalling:
  - If the AP provides features related to User Interface, the informative CPCM-USI may be used through the mapping specified in TS 102 825-10 [i.28].
  - The signal may be simply pushed further. For instance, if the informative signalling delivered by the HE over a MPEG-2 TS uses the `cpcm_delivery_signalling_descriptor`, and the CPCM content is handled in the form of the MPEG-2 TS inside the CPCM system, the signal can just be kept as is and used by other devices.

In case of use of DVB-SI, the descriptor for informative signalling may be inserted in (see EN 300 468 [i.10]):

- SDT: This saves bandwidth. It applies to the case when a service is broadcasting events requiring the same protection level, such as a premium content movie service for which copy is never allowed.
- EIT: This is done in any other case when the protection level changes from one Content item to another. Due to overriding rules, USI may be signalled in EIT for one event on a service where SDT already carries a generic USI. This case would apply to a service for which copy is seldom allowed.

This case extends to the case where the Usage Rules secured by the CAS can only represent a subset of CPCM-USI (e.g. SVCA or RAR-related fields are not supported). Then, the CPCM-USI entering the HE can be restricted to USI not encompassing setting of the unsupported CPCM features. This can be agreed between the operator and the content provider.

10.1.3 CPCM-USI mapped from CAS rights

At the opposite of the assumption made in previous clause, it is supposed here the CPCM-USI are not delivered to the HE but are computed from the CAS-rights, i.e. the right controlling end-user access to content according to CAS own rules. A specific mapping needs to be defined and the case falls under the same handling than DRM protected content (see clause 10.2), no matter if the CAS-rights control the access to content beyond acquisition or not.

10.1.4 Transfer of CAS-rights through CPCM System

The CAS may rely on CPCM-compliant devices in the HN to convey its proprietary rights associated to an acquired event and allow them to reach a CAS-integrated device that is able to apply them. In order to make sure the CAS-rights are conveniently transmitted with the corresponding CPCM Content, the CAS may apply two different policies:

- The CAS may implement in the AP, in addition to the first mapping, a mapping of the received CAS-rights to a dedicated format in the CLC_private_data element of the CPCM_auxiliary_data structure. The CLC_private_data structure particularly suits CAS private data, including a CA_system_id field, while the encapsulating CPCM_auxiliary_data is under the same protection level than the CL; its digest being part of the CL. Then, when the content reaches the CAS-integrated device, the CAS-rights may be retrieved and applied trustfully.

- The CAS may rely on the trusted CAS-rights to CPCM USI mapping, and define an inverse mapping to retrieve the original rights, or a satisfactory version of them, and apply them when the content reaches the CAS-integrated device. This is sufficient if the semantics of the CAS-rights and the underlying business models are also supported by CPCM features such as rental mode.

The CAS-integrated device may be a CPCM-compliant device, part of the CPCM System that acquired the original event, or it can be a device to which a CPCM Export Point exports the corresponding CPCM Content. This CPCM Export Point itself needs to be integrated with the CAS so to be able to apply one of the two above mappings.

10.1.5 USI provision

In addition to USI mapping at Acquisition, the HE has the task to provision USI at a convenient rate. In case of event streaming, the AP guarantees smooth transition between events. If the CSK is changed with the starting event, smooth transition is ensured inside CPCM by making use of the Odd-Even transition mechanism, as explained in the different cases of clause 6.4.1. In turn, for this mechanism to be applicable, the HE ensures that the USI corresponding to an event are transmitted in advance, so that the AP has enough time to build and push the new CL. This is also required for smooth channel transition (see clause 6.4.4.8).

10.2 DRM Mapping

10.2.1 Scope

In this clause, DRM systems are considered to also encompass external CPS systems or systems for protecting content on recordable or pre-recorded media.

These guidelines apply only to DRM systems that are being used to deliver content authorised for handover to DVB-CPCM control or to DRM systems that are authorised by some CPCM C&R regime to receive content from DVB-CPCM.

10.2.2 Approval

It is necessary for the DRM to be recognised as a Trusted Source or as a Trusted or Controlled Export system by the CPCM compliance regime. This implies a negotiated agreement between the DRM and CPCM C&R regimes.

This agreement defines the correct mapping(s) of rights expressed in the DRM system to the appropriate CPCM Usage State Information values and auxiliary data.

Implementers of CPCM in DRM-enabled systems are required to ensure they comply with both the DRM and CPCM compliance regimes.

NOTE: Compliance rules and requirements determined for a DRM system, and for CPCM, always take precedence over what is recommended in the present document. For example, it may be necessary for the governing compliance body of an individual DRM system to authorise CPCM as trusted source for that DRM system.

10.2.3 Mapping Definitions

Considerations on mapping definitions are also given in TR 102 825-13 [i.5]. Definition of mappings between two DRM systems is a difficult task as no exact mapping is likely to exist. For most situations, this is likely to result in the loss of some usage rights for the user once the Export has occurred.

When exporting content marked CCNA from CPCM, it is advised to retain a copy of the content within CPCM, as allowed by the USI, since some allowed content usage may not be enabled in the export DRM.

Another way to compensate for the potential loss of rights is to substitute one constraint with another one to get a similar effect.

EXAMPLE: Some DRM systems may not support the concept of copy control, but they support device binding. The CPCM compliance body may decide that device binding is a reasonable approximation for copy control, and allow the moving of Copy No More content (or copying of Copy Once content) via the Export point to be bound to a single device.

Operators of content delivery systems deployed for delivery of DVB services should grant rights to users that correspond to the relevant mapping, defined by the CPCM compliance body, between the CPCM Usage State Information and the DRM rights expression. Otherwise there is a risk of ambiguity in mapping, which may lead to variations in interpretation by differing implementations. This is undesirable for reasons of content protection (threat of accidental loosening of restrictions) and usability (inconsistent user experience).

10.2.4 Secure hand-over

Hand-over of content between a DRM and CPCM takes place in a secure environment that meets the compliance and robustness requirements of both the governing DRM and CPCM compliance regimes, such that no unprotected content is exposed beyond the secured environment without explicit authorisation.

NOTE: Even if the actual mapping is done in a different device, a hand-over is made in the Acquisition Point or in the Export Point to pass control to the mechanism used to establish trust with that device. The definition of such a mechanism is outside of CPCM scope.

10.2.5 Resolving Ambiguity

In the event that no clear mapping exists between rights expressed in the DRM and corresponding CPCM Usage State Information, implementations are required to assume that the most restrictive contextually-applicable case should apply (as stated in TS 102 825-3 [i.7]).

Such ambiguities are perhaps most likely to occur in the deployment of new services or business models. Therefore implementations should, where possible, also allow for updates to the mapping rules, such that ambiguities of this kind can be resolved over time.

10.2.6 Pass-through of DRM Information

Implementations should include DRM content protection information, such as rights expression language, in the content; in CPCM Content License as Auxiliary Information, in other content metadata, or elsewhere, such that content can eventually be returned to the control of the original DRM system with identical rights to those originally granted. Such auxiliary information needs to be protected as defined by the governing DRM compliance body.

10.3 CPCM delivery signalling mapping

In many situations, CPCM receives its USI from a secure source within the Acquisition Point, such as a DRM or a CAS component. However, there are two situations where USI may be received by means of CPCM delivery signalling as per TS 102 825-4 [i.9].

Firstly, this may be for informational purposes, such as in EPG metadata for describing upcoming events. In this case, while the signalling may be useful for presenting a meaningful user interface, such signalling cannot be used to derive the actual USI, even if no other information is available. The rationale for this is that CPCM delivery signalling is not secured while CPCM USI are supposed to be determined from a secure source. Thus such content, whose only available USI information comes from CPCM delivery signalling, will not normally be Acquired into CPCM.

Secondly, CPCM delivery signalling may be used to deliver the actual usage rules. This approach may be used by a clear-to-air broadcaster. In this case, A CPCM C&R regime needs to decide to use CPCM delivery signalling as a source for USI, either as a secondary source when the primary source is temporarily unavailable or as the unique usage rules source.

EXAMPLE: A Free-To-Air broadcaster wanting to use broader usage rules than the set defined in Free-To-Air broadcast signalling may decide to use CPCM delivery signalling as a means to convey its usage rules.

Unless otherwise defined by the governing C&R regime, most of the CPCM USI will be identical to the rights conveyed in the CPCM delivery signalling. The only exception is the remote_access_delay usage rule that will be converted to remote_access_date. The Acquisition Point is able to compute from the time of Acquisition and the delay the expiration date for the RAR rule. When the CPCM delivery signalling also embeds a remote_access_date, the earlier of the two dates will be retained for the CPCM USI. The original RAR rule will be retained since the conditions are different: if remote_access_date_immediate is set, the whole content will be accessible as soon as remote_access_date occurs. Otherwise, remote access to content will become available on the basis of a moving window (see clause 5.3.6).

11 Extension Guidelines

11.1 Extensions designs

The CPCM specification includes a framework for creating extensions which allow the implementation of functionalities that are not defined in the present document. CPCM allows for two specific types of Extensions, which are as follows:

- CPCM Extensions that are standardised as defined in TS 102 825-14 [i.6]. Guidelines for a specific CPCM Extension are given in clause 11.
- Private Extensions that are not standardised and can be developed by any implementer. Guidelines for private extensions are to be defined by the extension developer.

CPCM and private extensions need to be allocated an identifier. CPCM Extension identifiers are referred in TS 102 825-14 [i.6]. Identifiers for private extensions are private_data_specifiers_id, as defined in EN 300 468 [i.10], and can be obtained from the DVB Office. This specification does not impose any constraint on the way theses extension identifiers are allocated.

The implementation of a given CPCM or private extension is by nature optional.

CPCM extensions and/or private extensions have to be developed based on the fact that CPCM Instances that do not implement the extension will ignore any data, signalling or message related to that extension. This means that if, for instance, a new USI is proposed, the extension developer may have to define a particular USI setting to be enforced by CPCM Instances that do not implement the extension and a different USI setting for the other CPCM Instances.

Several tools exist for an extension developer:

- Definition of new data structures. The implementer is free to define any new data structure that will be identified using the extension identifier. In the case where the developer would need to define several new data structures, it is advised to use the same structure as the structure adopted by this specification: each data structure is allocated an extension-dependant identifier fixed length data structure do not need to carry a length field. This data structure may for instance be a new USI; a new device feature, …
- Definition of new messages specific to the extension. This can be performed using messages defined in TS 102 825-4 [i.9]. The first element carried in this message is the extension identifier. The remaining is defined in the extension definition. It may include CPCM defined n data structures or extension defined data structure.

- Addition of extension elements in defined CPCM messages. These elements will be added in the space reserved for extension elements in each message, as defined in TS 102 825-4 [i.9], and identified using extension identifier. These extension elements maybe CPCM defined data structure or extension defined data structure. In the case where, several data elements would be added, the extension developer is advised to use the same structure as the structure adopted by CPCM: fixed order for mandatory elements followed by conditional elements (with an associated count) and optional elements (with an associated count).

- Addition of extension elements in CPCM Auxiliary Data: These elements will be added in the space reserved for extension elements in CPCM auxiliary Data, as defined in TS 102 825-4 [i.9], and identified using extension identifier. These extension elements maybe CPCM defined data structure or extension defined data structure. In the case where, several data elements would be added, the extension developer is advised to use the same structure as the structure adopted by CPCM: fixed order for mandatory elements followed by conditional elements (with an associated count) and optional elements (with an associated count).

- For CPCM extensions, a space is reserved in Certificate format for carrying CPCM extension information. This could also be used to discover CPCM Instances implementing the CPCM extension.

As an example, if a specific CPCM or private extension needs to be verified before Content Licence is sent, two mechanisms may be used:

- Addition of Challenge / Response messages dedicated to the extension. The message would be an extension-defined message so that only extension–aware will participate.

- Usage of CPCM Instance status enquiry protocol: extension specific elements may be added to the reply and/or request message to allow easy identification of CPCM Instances implementing the extension.

11.2 Play Count Extension

11.2.1 Introduction

Playcount extension has been developed to impact only implementation of Acquisition Points and Storage Entities. As AP and SE also advertise the content as 'viewable' and 'copy once' or 'copy no more', Consumption Points or Export Points can access the content as if the content was not play limited. Only the Acquisition Point or the Storage Entity is in charge of enforcing the extension.

Storage Entities that do not implement the extension will discover the content as 'Copy No More' content and will thus use naturally the Move Protocol to transfer the content when requested by the user. Only Storage Entities implementing the extension have however the ability to split the number of remaining plays.

11.2.2 PlayCount Data Element

Table 12: Playcount Auxiliary Data Example

Element Name	Element Value
element_counter	0x01
CPCM_auxiliary_data_element_identifier	0x08
CPCM_auxiliary_date_element_length	0x0006
CPCM_extension_id	0x0001
CPCM_extension_data_length	0x0002
CPCM_play_limit	0x05
CPCM_play_remaining	0x02

Table 12 shows an exemplary auxiliary data structure embedding only one element relative to the PlayCount Extension. This element shows the associated content was delivered with a Play Count limited to 5 and 2 plays are remaining out of these 5 plays.

11.2.3 PlayCount Protocols

Table 13: Playcount Conditional Element in Move Protocol Example

Element Name	Element Value
conditional_element_count	0x01
element_identifier	0x1A
element_length	0x0006
CPCM_extension_id	0x0001
CPCM_extension_data_length	0x0002
CPCM_play_limit	0x05
CPCM_play_remaining	0x02

Table 13 shows an exemplary conditional element structure embedding only one element relative to the PlayCount Extension. This element can be used during the Move protocol. This element shows that 2 plays are requested to be Moved while content had 5 permitted plays upon Acquisition.

Table 14: Playcount Conditional Element in Discovery

Element Name	Element Value
conditional_element_count	0x01
element_identifier	0x1A
element_length	0x0004
CPCM_extension_id	0x0001
CPCM_extension_data_length	0x0000

Table 14 shows an exemplary conditional element structure embedding only one element relative to the PlayCount Extension. This element can be used in *CPCM Enquiry Response* or *CPCM Discovery Response* messages. It shows that the CPCM Instance implements the PlayCount extension.

11.2.4 USI Enforcement

11.2.4.1 Addressing specific threats

As with 'Copy No More' or 'View Period Activated' marked content, as defined in clause 6.4.3.8, implementations have to prevent the ability to copy back Content Licences that would have been duplicated before one or more plays occurred. Otherwise, the limited playcount could be turned into an infinite playcount.

The same examples as those given in clause 6.4.3.8 allow the prevention of this threat.

11.2.4.2 Simultaneous use with SVCA

As stated in TS 102 825-14 [i.6], when the Playcount extension is used simultaneously with SVCA USI, the number of simultaneous views can exceed the value in SVC field but at the cost of one extra play count. This applies only for live content, as the SVCA restriction ceases to apply once the live stream has ended.

EXAMPLE: If content is acquired with a Playcount of 7 and a simultaneous view count of 2, the user will be able to consume this content simultaneously on 5 consumption points but this will be counted as 3 plays.

List of figures

List of tables

History

Document history		
V1.1.1	March 2011	Publication