ETSI TS 102 825-4 V1.1.1 (2008-07)

Technical Specification

Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 4: CPCM System Specification

Reference
DTS/JTC-DVB-222-4

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

Contents

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 Specification (TS) 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

Founded in September 1993, the DVB Project is a market-led consortium of public and private sector organizations in the television industry. Its aim is to establish the framework for the introduction of MPEG-2 based digital television services. Now comprising over 200 organizations from more than 25 countries around the world, DVB fosters market-led systems, which meet the real needs, and economic circumstances, of the consumer electronics and the broadcast industry.

The present document is part 1 of a multi-part deliverable. Full details of the entire series can be found in part 1 [i.4].

Introduction

CPCM is a system for Content Protection and 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 specifications 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 is the specification of the Digital Video Broadcasting (DVB) Content Protection and Copy Management (CPCM) System. It contains the generic specification applicable to all deployments of CPCM. Technology-specific adaptations for content formats, storage formats, home network ecosystems and application-specific physical interfaces are contained in TS 102 825-9 [i.3].

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific.

- For a specific reference, subsequent revisions do not apply.
- Non-specific reference may be made only to a complete document or a part thereof and only in the following cases:
  - if it is accepted that it will be possible to use all future changes of the referenced document for the purposes of the referring document;
  - for informative references.

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

For online referenced documents, information sufficient to identify and locate the source shall be provided. Preferably, the primary source of the referenced document should be cited, in order to ensure traceability. Furthermore, the reference should, as far as possible, remain valid for the expected life of the document. The reference shall include the method of access to the referenced document and the full network address, with the same punctuation and use of upper case and lower case letters.

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 indispensable for the application of the present document. For dated references, only the edition cited applies. For non-specific references, the latest edition of the referenced document (including any amendments) applies.

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

[2] IEEE List of Registered OUI.

NOTE: Available at http://standards.ieee.org/regauth/oui/index.shtml.

[3] ISO 3166-1: "Codes for the representation of names of countries and their subdivisions - Part 1: Country codes".

[4] ETSI ETR 162: "Digital Video Broadcasting (DVB); Allocation of Service Information (SI) codes for DVB systems".

2.2 Informative references

The following referenced documents are not essential to the use of the present document but they assist the user with regard to a particular subject area. For non-specific references, the latest version of the referenced document (including any amendments) applies.

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

[i.4] ETSI TS 102 825-1: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 1: CPCM Abbreviations, Definitions and Terms".

[i.5] ETSI TS 102 825-2: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 2: CPCM Reference Model".

[i.6] ETSI TR 102 825-8: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 8: CPCM Authorized Domain Management scenarios".

[i.7] ETSI TR 102 825-11: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 11: CPCM Content management scenarios".

[i.8] ETSI TS 102 825-5: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 5: CPCM Security Toolbox".

[i.9] ETSI TS 102 825-7: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 7: CPCM Authorized Domain Management".

[i.10] 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.11] ETSI TS 102 825-3: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 3: CPCM Usage State Information".

[i.12] ETSI TR 102 825-12: "Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 12: CPCM Implementation Guidelines".

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.4] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 102 825-1 [i.4] apply.

4 The CPCM System

4.1 General

The CPCM System provides an interoperability platform for the protection and management of "commercial" content, i.e. not user-created content, in the consumer environment.

The CPCM Reference Model (TS 102 825-2 [i.5]) and the CPCM Abbreviations, Definitions and Terms (TS 102 825-1 [i.4]) provide the foundation upon which this CPCM System specification is based. The CPCM System specification populates each of the aspects of the CPCM Reference Model in order to define a complete and implementable system.

The main body of this document provides the generic specification of the CPCM System, applicable to all deployments of CPCM.

The generic CPCM System specification defines the aspects of the CPCM System that apply to all deployments of CPCM. Subsequent sub-sections of this clause deal with the following elements of the CPCM System:

- CPCM Security Toolbox;
- CPCM Authorized Domain Management (ADM);
- CPCM System versioning;
- CPCM Device physical interfaces; and
- CPCM Instance logical interfaces.

The remaining sections of the CPCM System generic specification, starting in clause 5, deal with the following elements:

- **CPCM data formats:** the generic CPCM System specification contains a common set of CPCM-specific data type definitions used by every CPCM System deployment. The CPCM Data Types are documented in clause 5.
- **CPCM protocols:** the generic CPCM System specification defines a set of protocols that are used by all CPCM Instances in order to enable CPCM Content exchanges and exchange CPCM-relevant data. The CPCM protocols are specified in clause 6.
- **CPCM Content management:** individual aspects of CPCM System behaviour relating to the handling of CPCM Content are specified in clause 7.
- **CPCM Content Licence management:** clause 8 deals with all aspects around the generation and handling of CPCM Content Licences.

Various end-to-end CPCM Content handling scenarios, as well as ADM scenarios, are documented in TR 102 825-8 [i.6] and TR 102 825-11 [i.7], building on the individual content handling aspects from clauses 7 and 8.

The CPCM System is designed to provide a generic content protection and management platform that can be adapted to various content ecosystems. A content ecosystem can consist of one or more aspects to which the CPCM System Specification is adapted in order to provide a complete and workable content protection and management solution. These aspects of adaptation are:

- **Content format:** a CPCM System deployment needs at least one content format adaptation. CPCM System content format adaptations are defined in TS 102 825-9 [i.3].
- **Storage format:** if a CPCM System deployment includes the ability to Store CPCM Content then it needs at least one storage format. CPCM System storage format adaptations are defined in TS 102 825-9 [i.3].
- **Home Network ecosystem:** a home network ecosystem adaptation is needed for the CPCM System to work within that ecosystem. Specifications of CPCM System adaptations for particular home network ecosystems are contained in TS 102 825-9 [i.3].

- **Application-specific physical interface:** the CPCM System can be adapted to provide content protection and management functionality over an application-specific physical interface that benefits from re-using the CPCM tools and facilities. TS 102 825-9 [i.3] contains the definitions for CPCM System adaptations for particular application-specific physical interfaces.

4.2 CPCM Security toolbox

The CPCM Security Toolbox is the set of cryptographic algorithms and methods that enables the protection of CPCM Content, as required by the provider of that content, throughout its lifetime within the CPCM System.

The set of CPCM security tools in the toolbox do not provide any alternative methods for performing any particular security-related task within the CPCM System. Rather, there is one tool specified for each security-related task.

A CPCM Instance shall implement all security tools defined in the CPCM Security Toolbox that are necessary to perform its CPCM functionality. A CPCM Instance that is intended to have access to the full range of CPCM Content must naturally implement the complete CPCM Security Toolbox.

The CPCM Security Toolbox is contained in a separate part of the present document (TS 102 825-5 [i.8]).

4.3 CPCM Authorized Domain Management

CPCM Authorized Domain Management (ADM) comprises the rules and principles of operation of Authorized Domains in the CPCM System. It defines how CPCM Authorized Domains are created and administered, and how CPCM Devices and CPCM Instances relate to the concept of the CPCM Authorized Domain and manage their AD membership, if applicable.

The implementation of ADM in a CPCM Instance is optional. If the CPCM Device hosting that CPCM Instance is not AD aware then the implementation of ADM is not required in that CPCM Instance. CPCM Devices/Instances that are not AD aware naturally have restricted usability of CPCM Content that is bound to an AD, or of Input Content that needs to be bound to an AD.

CPCM Authorized Domain Management is defined in TS 102 825-7 [i.9].

Aspects related to the management of CPCM Content within an Authorized Domain or between different ADs are documented in Clause 8 of CPCM Content Licence management.

4.4 CPCM System versioning

CPCM System versioning is facilitated by the maintenance of a global CPCM_version parameter, stored in the CPCM Instance Certificate, in the CPCM Content Licence, in the CPCM Usage State Information (USI) structure, and in the CPCM Revocation List data structures.

CPCM System deployments that comply with the present specification shall set CPCM_version to 1.

CPCM_version is an 8-bit field so that the CPCM System specification could be revised up to version 255.

An increment of the value of CPCM_version shall be applied if any of the following elements of the CPCM System is changed:

- CPCM Instance Certificate (CIC) format;
- CPCM Content Licence (CL) format;
- CPCM Content Revocation List (RL) format;
- CPCM Usage State Information (USI) format;
- CPCM delivery signalling; or
- CPCM protocol syntax.

Potential future revisions of the CPCM System specification should aim to provide backwards compatibility with the present document.

Implementations of CPCM Instances shall interoperate with CPCM Instances supporting previous versions of CPCM if the functionality provided by each of the CPCM Instances makes their interoperation feasible and useful, for example with respect to the handled content formats.

Higher-version CPCM Instances shall, during their discovery process, make themselves aware of any lower-version CPCM Instances that may be present, so that the higher-version CPCM Instance shall correctly interpret received lower-version CPCM protocol communications when a lower-version CPCM Instance initiates CPCM protocol communications with the higher-version CPCM Instance.

When a higher-version CPCM Instance initiates CPCM protocol communications with a lower-version CPCM Instance then it shall revert to the CPCM protocol format understood by the lower-version CPCM Instance, if the CPCM protocol format has changed in the higher version.

The structure of the CPCM Instance Certificate (CIC), specified in clause 5.4.2, accommodates the possibility to add backwards-compatible extensions, so that lower-version CPCM Instances will be able to work with higher-version CICs but will just not be able to work with the additional fields.

Regarding revisions to the CPCM Content Licence format, if CPCM Content Acquired by the higher-version CPCM System deployment is intended to be usable for the lower-version CPCM Instances, then the higher-version CPCM Instances shall be able generate Content Licences that are compliant with all versions of CPCM for which the CPCM Content is intended to be usable, and exchange only the appropriate version of Content Licence with the lower-version CPCM Instances.

4.5 CPCM Device physical interfaces

The generic part of the CPCM System Specification does not specify any particular CPCM Device physical interfaces. Rather, they are specified either implicitly or explicitly in the CPCM System adaptations. For this purpose the CPCM System Specification makes the distinction between two types of CPCM Device physical interface:

- **Generic home network ecosystem device interface:** this is one or more interfaces specified by the home network ecosystem in order to facilitate interoperability of devices in a home network that provides general device control and content exchange functionality, as well as the handling of CPCM Content (as specified by the present specification). Specifications of the CPCM System adaptations for particular home network ecosystems are contained in TS 102 825-9 [i.3].

- **Application-specific physical interface:** this is a physical interface for which the present document specifies the mode of operation of CPCM over that interface as a dedicated extension to the functionality of the interfaces for that purpose alone, i.e. not to provide interoperability with any wider ecosystem. Application-specific physical interfaces extend the reach of the CPCM System beyond those physical interfaces defined within any home network ecosystem. TS 102 825-9 [i.3] defines the specifications of the CPCM System adaptations for particular application-specific physical interfaces. Further application-specific physical interfaces may be specified by a CPCM System compliance regime, for example for CPCM Device Consumption Output interfaces. Such specifications are out of scope for the present specification.

4.6 CPCM Instance Logical Interfaces

Figure 1 shows the set of CPCM Device logical interfaces, mapped from the CPCM Reference Model, applied to two hypothetical CPCM Devices for the purpose of illustration. CPCM Device physical interfaces are differentiated as described in clause 4.5. A nominal generic home network physical interface is shown in a lighter shade of grey and hypothetical application-specific physical interfaces are shown in a darker shade.

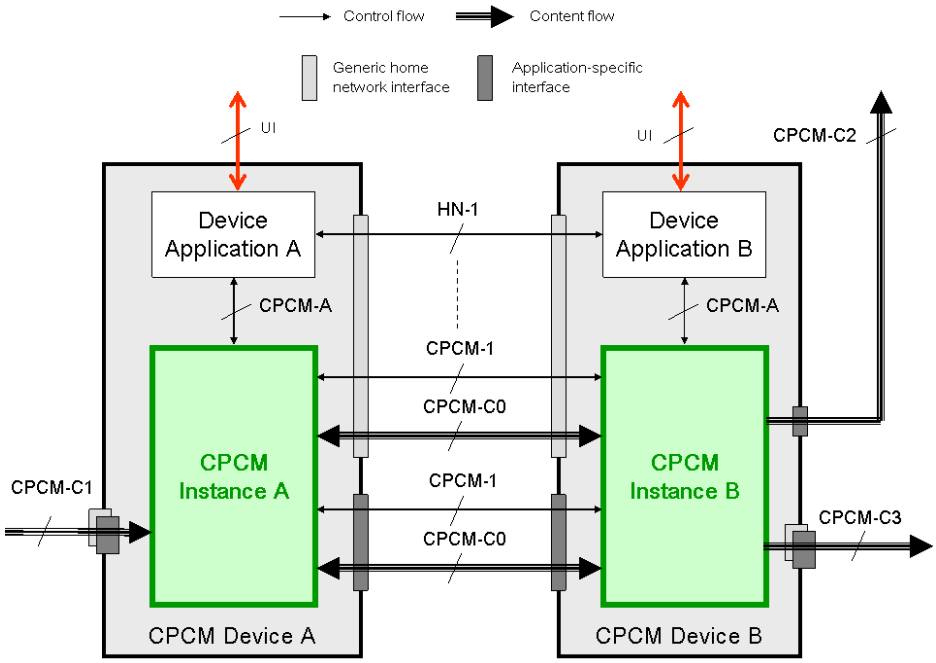

Figure 1: CPCM Generic Device Interfaces

The CPCM logical interfaces are divided into those for control (thin arrows) and those for content flows (thick arrows). The User Interface (UI) is a special case that is elaborated, but where no actual specification is undertaken.

Figure 2 shows the logical interfaces of the CPCM Instance, including the informative intra-CPCM Instance interfaces CPCM-I1 and CPCM-I2, corresponding to CPCM Control and ADM Control interfaces respectively, in the CPCM Reference Model.

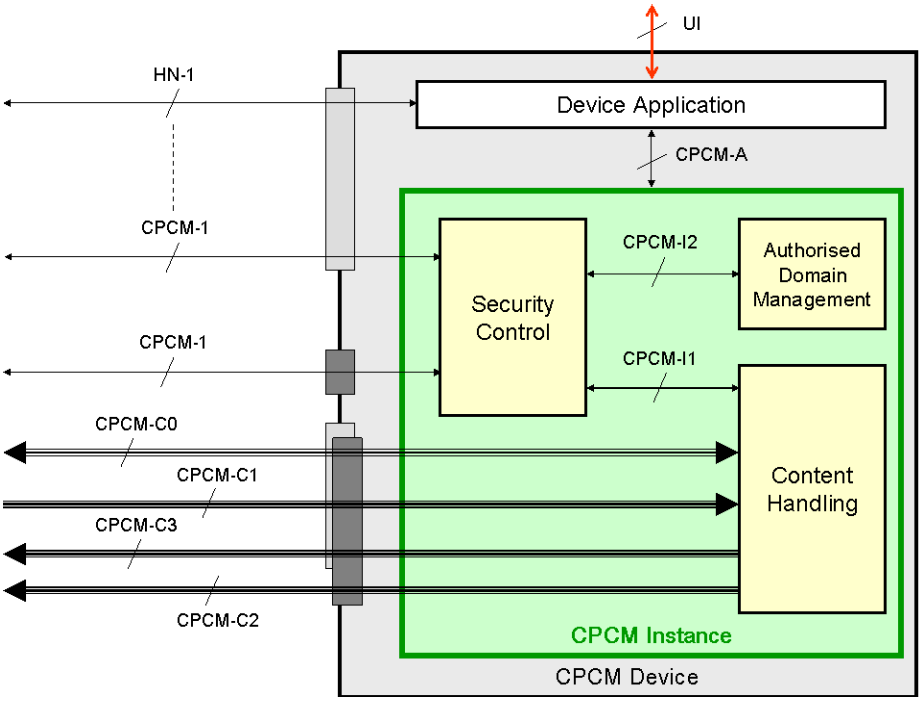

Figure 2: CPCM Device and Instance Logical Interfaces

Table 1 lists the set of CPCM System logical interfaces and indicates which are specified normatively in this document, and which are documented informatively. The generic aspects of each of the logical interfaces are detailed below.

Clause 5 defines the CPCM protocols provided for the normatively specified logical interfaces.

Table 1 CPCM Instance Logical Interfaces

CPCM Instance logical interface	CPCM Reference Model mapping	Notes
HN-1	Device Interaction, CPCM Discovery Communications, ADM Non-Secure Communications	Normative
CPCM-1	CPCM Trust Establishment, CPCM Communications, Proximity Control Communications	Normative
CPCM-A	CPCM Interaction	Informative
UI	User Interface	Informative
CPCM-I1	CPCM Control	Informative
CPCM-I2	ADM Control	Informative
CPCM-C0	Content flow of C.0 CPCM Content	Normative
CPCM-C1	Content flow of C.1 Input Content	Normative
CPCM-C2	Content flow of C.2 Consumed Content	Normative
CPCM-C3	Content flow of C.3 Exported Content	Normative

Further notes on the CPCM Instance logical interfaces:

HN-1:

Interface HN-1 corresponds to the Device Interaction logical interface in the CPCM Reference Model. This is the home network ecosystem-specific set of protocols that enable home network device discovery interaction, and content discovery and usage between devices on the home network.

CPCM Discovery enables CPCM Instances to become aware of each other's presence in the home network, and to become aware of the intransient CPCM attributes of the other CPCM Devices and Instances.

The CPCM Device discovery process shall either be an integral part of the home network device discovery process, for example by adding CPCM-specific attributes to the device attributes advertised in the discovery process, or it shall take place as a separate discovery process. If a CPCM-specific discovery process is integrated within a particular home network ecosystem then it is defined in the corresponding adaptation layer in TS 102 825-9 [i.3]. Otherwise CPCM-specific discovery is performed using the general mapping of the CPCM protocol messages to that ecosystem, which is also defined in the corresponding adaptation layer in TS 102 825-9 [i.3].

This interface definition is normative. The corresponding CPCM protocols are specified generically in clause 6.

CPCM-1:

This is the core logical interface for CPCM, encapsulating the set of CPCM protocols defined in clause 6.

The CPCM-1 interface contains the following elements in terms of mapping from the logical interfaces defined in the CPCM Reference Model:

- CPCM Security Control (SEC) protocols, consisting of the elements:
  - CPCM Authenticated Key Exchange (AKE); and
  - AD Secret Exchange.
- CPCM System and Content Management (SYS) protocols, consisting of the elements:
  - CPCM discovery protocol;
  - CPCM Instance status exchange;
  - CPCM Content item status enquiry;
  - CPCM Content operation protocol;
  - CPCM Content Licence exchange;
  - CPCM Revocation List acquisition;
  - CPCM Instance AD membership verification;
  - CPCM proximity control;
  - CPCM Content move protocol;
  - Secure time exchange; and
  - Geographic information exchange.
- Authorized Domain Management (ADM) protocols, consisting of the elements:
  - AD discovery, AD membership management (join, leave);
  - ADM Local Master and Domain Controller management, as defined in TS 102 825-7 [i.9].

CPCM-A:

This is the internal CPCM Device interface between the device's native application and the CPCM Instance. This interface is informative, but having certain derived minimum requirements for the information to be conveyed:

- CPCM-specific content discovery, e.g. information about the Authorized Usage;
- Content usage requests and feedback about their status;
- AD Management for the device.

UI:

Informative, but having certain minimum requirements for the information to be conveyed.

Informative requirements:

- Convey information about the Authorized Usage of individual Content items;
- Communicate to the user in case CPCM Content revocation has been applied on the CPCM Device;
- Interaction concerning AD Management, if applicable for that device.

CPCM-I1 and CPCM-I2:

The intra-CPCM Instance interfaces CPCM-I1 and CPCM-I2 are informational as it is completely at the discretion of the manufacturer how these are implemented inside a CPCM Device.

CPCM-C0:

The normative generic specification of the handling of CPCM Content is contained in clause 7 on CPCM Content Management.

The format of CPCM Content is one of the adaptation layers of the CPCM System specification contained in TS 102 825-9 [i.3].

CPCM-C1:

The normative generic specification of the handling of CPCM Input Content (C.1) upon Acquisition into the CPCM System is contained in clause 7 on CPCM Content Management.

The normative specifications of how particular types of Input Content are Acquired are contained in TS 102 825-10 [i.10].

CPCM-C2:

The normative generic specification of the handling of the Consumption of CPCM Content is contained in clause 7 on CPCM Content Management.

It is not foreseen to transport Consumed Content (C.2) over a generic home network ecosystem physical interface.

Standardized specifications of physical interfaces for the carriage of Consumed Content at Device Outputs are contained in TS 102 825-10 [i.10]. Such mappings can also be defined by the C&R regime.

CPCM-C3:

The normative generic specification of the handling of the Export of CPCM Content is contained in clause 7 on CPCM Content Management.

Standardized specifications of both physical interfaces for the transmission of Exported Content at Device Outputs, and storage formats for the carriage of Exported Content are contained in TS 102 825-10 [i.10]. Such mappings can also be defined by the C&R regime.

5 CPCM Data Types

5.1 General

This clause contains the normative definitions of the set of data types required for the CPCM System.

These definitions are valid for all deployments of the CPCM System, i.e. all adaptations to home network ecosystems and application-specific physical interfaces shall implement whichever of these data type definitions that are necessary for that adaptation.

5.2 Basic types

5.2.1 CPCM witness

CPCM_witness is a 1024-bit number used for CPCM Trust Establishment and AKE. The method of calculation of this parameter is defined in TS 102 825-5 [i.8].

5.2.2 CPCM trust check

CPCM_trust_check is a 160-bit number used for CPCM Trust Establishment and AKE. The method of calculation of this parameter is defined in TS 102 825-5 [i.8].

5.2.3 CPCM Message Authentication Code (MAC)

CPCM_MAC (Message Authentication Code) is a 160-bit number used to authenticate CPCM protocol messages. A CPCM MAC can be derived either from the SAC session key or from the AD Secret. The algorithm applied for the generation of MAC is defined in TS 102 825-5 [i.8].

5.2.4 CPCM AD secret

CPCM_AD_secret is a 128-bit number that is the unique, permanent secret for the Authorized Domain.

5.2.5 Signed CPCM Instance Certificate

Signed_CPCM_Instance_Certificate is a 2048-bit number representing the CPCM Instance Certificate, as defined in TS 102 825-5 [i.8].

5.2.6 CPCM playback period

CPCM_playback_period is a 16-bit number representing a number of 15-minute periods.

5.3 CPCM identifiers

5.3.1 CPCM Instance identifier

The CPCM_instance_id is a 64-bit number assigned to each CPCM Instance by the manufacturer. Its purpose is to securely bind the CPCM Instance Certificate to its CPCM Instance.

Device manufacturers can freely assign CPCM Instance identifiers, but each CPCM Instance identifier shall be unique.

5.3.2 CPCM Certificate identifier

The CPCM Certificate Identifier is a unique number assigned to each CPCM Certificate associated with a CPCM Instance, based on the IEEE 64-bit Extended Unique Identifier (EUI-64), which consists of the 24-bit company_id of the issuer of the CPCM Certificate, assigned by the IEEE Registration Authority committee (RAC), plus a 40-bit individual identifier, assigned by the issuer of that CPCM Certificate.

5.3.3 CPCM Authorized Domain Identifier

The CPCM Authorized Domain Identifier (ADID) is a 72-bit unique number used to identify an Authorized Domain.

The method of ADID allocation is defined in TS 102 825-7 [i.9].

The value of ADID set to zero is reserved to mean "not an AD member" so this value shall not be assigned as an ADID.

5.3.4 CPCM Content Licence Identifier

CPCM_content_licence_identifier (CLID) is a 128-bit number that provides 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. Since there is no global registration foreseen for CPCM Content Licence Identifiers, the link to the CPCM Content item shall be maintained by an appropriate allocation of the CPCM Content Licence Identifier by the Acquisition Point generating that CPCM Content Licence, by the application of the chosen CLID allocation scheme and the corresponding setting of the scheme-specific Content Licence identifier. The CLID allocation schemes defined by the present document are described in clause 8.3.

The format of CPCM_content_licence_identifier is shown in table 2.

Table 2: CPCM Content Licence Identifier

Syntax	Bits	Identifier
CPCM_content_licence_identifier(){		
CLID_allocation_scheme	8	uimsbf
scheme_specific_CL_identifier	120	uimsbf
}		

Semantics for CPCM_content_licence_identifier:

CLID_allocation_scheme: This field indicates which CLID allocation scheme has been applied upon Acquisition of the corresponding Content item. The CLID allocation schemes are specified in clause 8.2.1.5.

scheme_specific_CL_identifier: This field is the Content Licence Identifier that provides the link to the associated CPCM Content item within the applied CLID allocation scheme. The format of scheme_specific_CL_identifier within each CLID allocation scheme is specified in clause 8.2.1.5.

5.4 CPCM data structures

5.4.1 CPCM date and time

CPCM_date_time is a structure containing an absolute date and time setting. The format is as defined for use in DVB Service Information (EN 300 468 [1]). Table 3 shows the format of CPCM_date_time based on this definition.

Table 3: CPCM Date and Time

Syntax	Bits	Identifier
CPCM_date_time(){		
MJD	16	uimsbf
UTC	24	uimsbf
}		

Semantics for CPCM_date_time:

MJD: This is the Modified Julian Date, as used in DVB-SI (EN 300 468 [1]).

UTC: This is the Universal Time, Co-ordinated, as used in DVB-SI (EN 300 468 [1]).

For example 93/10/13 12:45:00 is coded as "0xC079124500".

5.4.2 CPCM Instance Certificate Body

Table 4 shows the format of the CPCM Instance Certificate body.

The total length of the CPCM Instance Certificate body is 896 bits.

Table 4: Unsigned CPCM Instance Certificate

Syntax	Bits	Identifier
CPCM_instance_certificate_body(){		
CPCM_version = 0x01	8	uimsbf
CPCM_instance_id	64	bslbf
CPCM_instance_certificate_id	64	bslbf
issuer_id	64	bslbf
C_and_R_regime_mask	8	bslbf
certificate_expiration_time	40	CPCM_date_time
generation_index	8	uimsbf
is_signer	1	bslbf
is_revocation	1	bslbf
reserved	1	bslbf
content_handling_capability	5	bslbf
AD_aware	1	bslbf
ADM_capable	1	bslbf
ADM_LM_capable	1	bslbf
ADM_DC_capable	1	bslbf
ADSE_countable	1	bslbf
LSA_capable	1	bslbf
absolute_time_capable	1	bslbf
geographic_aware	1	bslbf
reserved	624	bslbf
}		

Semantics for the unsigned CPCM_instance_certificate_body:

CPCM_version: CPCM System version supported by the CPCM Instance. CPCM Instances supporting the present document shall set CPCM_version to 0x01. CPCM_version may increment with later revisions of the present document.

CPCM_instance_id: Unique identifier for the CPCM Instance, used for the purpose of binding the CPCM Certificate to the CPCM Instance. The CPCM_Instance_ID is generated by the CPCM Instance manufacturer.

CPCM_instance_certificate_id: Unique identifier for the CPCM Instance Certificate (CIC), used for purposes of CPCM Instance identification in the home network ecosystem, certificate revocation and for AD management. The certificate identifier is generated by the CPCM Certificate provider.

issuer_id: Unique identifier of the certificate that was used to sign this certificate.

C_and_R_regime_mask: Identifies which of the 8 C&R regimes are supported.

certificate_expiration_time: The time in CPCM time (minutes from CPCM time zero) after which a certificate is considered expired. This field is checked to enforce USI but not checked during trust establishment. A value of zero means that the certificate never expires.

generation_index: Time-independent parameter for certificate renewal.

is_signer: Whether the certificate is a Signer certificate, i.e. its public key is used for verifying the signature of other Signing offspring certificates or a device certificate.

is_revocation: Whether the certificate is the revocation certificate of one of the C&R regimes.

content_handling_capability: Bit mask for the CPCM Functional Entities implemented in the CPCM Instance, with 1 bit for each of APECS, in that order.

AD_aware: Whether the CPCM Instance is able to enforce AD-related USI.

ADM_capable: Whether the CPCM Instance is capable of performing ADM functionality.

ADM_LM_capable: Whether the CPCM Instance can act as an ADM Local Master or not.

ADM_DC_capable: Whether the CPCM Instance can act as an ADM Domain Controller or not.

ADSE_countable: Whether the CPCM Instance shall affect ADM ADSE counts or not.

LSA_capable: Whether the CPCM Instance is capable of scrambling and/or descrambling CPCM Content (using LSA).

absolute_time_capable: Whether the CPCM Instance is capable of maintaining secure absolute time.

geographic_aware: Whether the CPCM Instance is capable to know is geographic location.

The **reserved** bits shall be set to zero. Future versions of the CPCM specification may utilize the reserved bits.

5.4.3 CPCM Instance Certificate chain

CPCM_instance_certificate_chain is a variable length structure containing, in principle, a concatenation of up to 15 chained CPCM Instance Certificates (CIC). Its structure shall be such that the first CIC contained is the leaf certificate and each further certificate contained is the parent certificate of the preceding certificate. The last certificate contained shall be the one signed by the root certificate. Its structure is shown in table 5.

Table 5: CPCM Instance Certificate Chain

Syntax	Bits	Identifier
CPCM_instance_certificate_chain(){		
reserved	4	uimsbf
certificate_count	4	uimsbf
for(i=0; i<certificate_count; i++) {		
CPCM_instance_certificate	2048	signed_CPCM_instance_certificate
}		
}		

Semantics for the CPCM_instance_certificate_chain:

The **reserved** bits shall be set to zero. Future versions of the CPCM specification may utilize the reserved bits.

certificate_count: The number of CPCM Instance Certificates contained in the chain.

CPCM_instance_certificate: Each Signed CPCM Instance Certificate in the chain for the corresponding CPCM Instance.

5.4.4 CPCM Content Licence

5.4.4.1 General

The CPCM Content Licence is a variable-length structure, with three possible optional fields, as shown in table 6. Due to the variability in length of the mandatory fields RL_index_list and usage_state_information, the length field is included to enable easier parsing of the Content Licence from the end of its contents.

Table 6: CPCM Content Licence

Syntax	Bits	Identifier
CPCM_content_licence(){		
CPCM_version	8	uimsbf
length	8	uimsbf
content_licence_protection	1	bslbf
descrambler_information	7	bslbf
content_licence_identifier	128	bslbf
content_licence_creator	64	bslbf
last_CL_issuer	64	bslbf
C_and_R_regime_mask	8	bslbf
RL_index_list()	N * 32	N * uimsbf
AD_identifier	72	bslbf

Syntax	Bits	Identifier
descrambling_key	128	bslbf
usage_state_information()	see clause 5.4.4.3	CPCM_usage_state_information
[CPCM_auxiliary_data_digest]	160	bslbf
[AD_secret_signature]	160	bslbf
}		

Semantics for the CPCM_content_licence:

CPCM_version: CPCM System version supported by the CPCM Instance. CPCM Instances supporting the present document shall set CPCM_version to 0x01. CPCM_version may increment with later revisions of the present document.

length: The number of bytes following the length field in the Content Licence, including any of the optional fields.

content_licence_protection: This field indicates how the CL is protected. A value of 1 means that the CL is signed using the AD Secret and the descrambling_key (if applicable) is encrypted using the AD Secret. A value of 0 means that no such explicit signing and descrambling_key encryption is applied and the CL shall be exchanged between CPCM Instances within the CPCM SAC, and stored securely by the CPCM Instance by other means. An exception may be that for CPCM Content for which the do_not_cpcm_scramble bit is set in the USI (see clause 5.4.4.3; so that the content is transferred without the application of the CPCM Scrambler), the CL is also exchanged in the clear, i.e. no SAC needs be in place. This exception shall be applicable at the discretion of the C&R regime.

descrambler_information: A structure containing CPCM Descrambler parameters (TS 102 825-5 [i.8]) set by the CPCM Device that Sources the associated CPCM Content item. The format of this field is defined in clause 5.4.4.2.

content_licence_identifier (CLID): Identifier of the CPCM Content Licence, providing the link to the associated CPCM Content item.

content_licence_creator (CLC): CPCM Certificate Identifier of the CPCM Instance that Acquired this CPCM Content item into the CPCM System.

last_CL_issuer: This field contains the CIC identifier of the CPCM Instance that last issued the Content Licence. Its use is specified in clause 8.

C_and_R_regime_mask: This field identifies under which of the 8 C&R regimes the associated CPCM Content item is covered.

RL_index_list(): This is the list of CPCM Revocation List index, one RL_index entry per bit set in the C_and_R_revocation mask field, in the order from msb to lsb.

AD_identifier (ADID): ADID of the AD to which the CPCM Content item is bound, if applicable. If the associated CPCM Content item is not bound to an AD then this field shall be set to zero.

descrambling_key: CPCM Descrambling key for the associated CPCM Content item (TS 102 825-5 [i.8]).

usage_state_information (USI): This is the CPCM USI applying to the associated CPCM Content item. The format of this field is specified in clause 5.4.4.3.

CPCM_auxiliary_data_digest: This field, if present, contains the signature of the CPCM Auxiliary Data which is embedded in the CPCM Content item referred to by the Content Licence. This field is present only when there is CPCM auxiliary data associated with the Content item. CPCM Auxiliary Data is specified in clause 5.4.9.

AD_secret_signature: This field contains the signature (i.e. the Message Authentication Code) of the CL, calculated using the AD Secret, before encryption of the descrambling_key field. This field is present only when the content_licence_protection field is set to 1.

5.4.4.2 CPCM Descrambler information

The CPCM Descrambler Information is a 7-bit field set by the Acquisition Point to indicate which CPCM Scrambler parameters have been applied to the associated CPCM Content item. This information is necessary for content descrambling. Table 7 shows the format of the CPCM Descrambler Information.

Table 7: CPCM Descrambler Information

Syntax	Bits	Identifier
CPCM_descrambler_information(){		
odd_even_key_indicator	1	bslbf
reserved	4	bslbf
descrambler_chaining_mode	1	bslbf
descrambler_MSC_mode	1	bslbf
}		

Semantics for CPCM_descrambler_information:

odd_even_key_indicator: Where applicable, indicator for which Descrambling key is applicable to the associated CPCM Content item, 0 = Even, 1 = Odd. The usage of this field is described in clause 7.

descrambler_chaining_mode: Chaining Mode applied to the associated CPCM Content item (TS 102 825-5 [i.8]).

descrambler_MSC_mode: MSC Mode applied to the associated CPCM Content item (TS 102 825-5 [i.8]).

The **reserved** bits shall be set to zero. Future versions of the CPCM specification may utilize the reserved bits.

5.4.4.3 CPCM Usage State Information

CPCM Usage State Information (USI) is CPCM Content metadata that signals the Authorized Usage for each CPCM Content item. The set of CPCM USI is defined in detail in TS 102 825-3 [i.11]. Table 8 shows the structure of CPCM USI.

Table 8: CPCM USI

Syntax	Bits	Identifier
CPCM_usage_state_information(){		
length	8	uimsbf
copy_control	3	uimsbf
do_not_cpcm_scramble	1	bslbf
viewable	1	bslbf
view_window_activated	1	bslbf
view_period_activated	1	bslbf
simultaneous_view_count_activated	1	bslbf
move_local	1	bslbf
view_local	1	bslbf
move_and_copy_propagation_information	2	uimsbf
view_propagation_information	2	uimsbf
remote_access_date_moving_window_flag	1	bslbf
remote_access_date_immediate_flag	1	bslbf
remote_access_record_flag	1	bslbf
export_controlled_cps	1	bslbf
export_beyond_trust	1	bslbf
disable_analogue_sd_export	1	bslbf
disable_analogue_sd_consumption	1	bslbf
disable_analogue_hd_export	1	bslbf
disable_analogue_hd_consumption	1	bslbf
image_constraint	1	bslbf
if (view_window_activated == 1){		
view_window_start	40	CPCM_date_time
view_window_end	40	CPCM_date_time
}		
if (view_period_activated == 1){		
view_period_from_first_playback	16	CPCM_playback_period
}		
if (simultaneous_view_count_activated == 1){		
simultaneous_view_count	8	uimsbf
}		
if ((remote_access_date_immediate_flag == 1) || (remote_access_date_moving_window_flag == 1)){		

Syntax	Bits	Identifier
remote_access_date	40	CPCM_date_time
}		
if (export_controlled_cps == 1){		
cps_vector_count	8	uimsbf
for (i=0; i<cps_vector_count; i++) {		
C_and_R_regime_mask	8	bslbf
cps_vector_length	16	uimsbf
for (j=0; j<cps_vector_length; j++){		
cps_vector_byte	8	bslbf
}		
}		
}		
}		

Semantics for CPCM_usage_state_information:

length: This field indicates the number of bytes following the length field in the CPCM_usage_state_information structure.

copy_control: This 3-bit field indicates the copy control information. It shall be coded according to table 9.

Table 9: Copy control

cci_and_zero_retention	Description
0	Copy Control Not Asserted
1	Copy Once
2	Copy No More
3	Copy Never - Zero Retention Not Asserted
4	Copy Never - Zero Retention Asserted
5 to 7	Reserved for future use

do_not_cpcm_scramble: This 1-bit field indicates whether or not to apply CPCM scrambling of the content item. If set to '1' then scrambling shall not be applied. If set to '0' then scrambling shall be applied.

viewable: This 1-bit field indicates that viewing, i.e. Consumption and Export of the content item are enabled.

view_window_activated: This 1-bit field indicates that viewing, i.e. Consumption and Export are limited by the view_window_start and view_window_end fields.

view_period_activated: This 1-bit field indicates that viewing, i.e. Consumption and Export are limited by the view_period_from_first_playback field.

simultaneous_view_count_activated: This 1-bit field indicates that simultaneous viewings, i.e. Consumptions and Exports are numerically limited by the simultaneous_view_count field.

move_local: This 1-bit field indicates that copying and/or Movement is allowed if the destination is local to the source, e.g. both are within the same home. This field is equivalent to the term MLocal used in TS 102 825-3 [i.11].

view_local: This 1-bit field indicates that viewing, i.e. consumption is allowed if the consumption point is local to the source, e.g. both are within the same home. This field is equivalent to the term VLocal used in TS 102 825-3 [i.11].

move_and_copy_propagation_information: This 2-bit field indicates the Move and Copy Propagation Information. It shall be coded according to table 10.

Table 10: Movement and Copying propagation information

Move and Copy propagation information	Description
0	MLAD - Copying and/or Movement within the same Localized AD is allowed.
1	MGAD - Copying and/or Movement within the same Geographically-constrained AD is allowed.
2	MAD - Copying and/or Movement within the same Authorized Domain is allowed.
3	MCPCM - Copying and/or Movement to any CPCM-compliant Storage Entity is allowed.

view_propagation_information: This 2-bit field indicates the Viewing Propagation Information. It shall be coded according to table 11.

Table 11: Viewing propagation information

View propagation information	Description
0	VLAD - Viewing, i.e. Consumption, within the same Localized AD is allowed.
1	VGAD - Viewing, i.e. Consumption, within the same Geographically-constrained AD is allowed.
2	VAD - Viewing, i.e. Consumption, within the same Authorized Domain is allowed.
3	VCPCM - Viewing, i.e. Consumption, using any CPCM-compliant Consumption Point is allowed.

remote_access_date_moving_window_flag: This 1-bit field indicates that remote access is permitted on the basis of a real-time moving window from the time and date indicated by the remote_access_date field.

remote_access_date_immediate_flag: This 1-bit field indicates that remote access to the complete Content item is permitted immediately from the date indicated by the remote_access_date field.

remote_access_record_flag: This 1-bit field indicates that remote access is permitted after the end of the delivery of the content item.

export_controlled_cps: This 1-bit field indicates that content may be exported or output to Controlled CPSs defined in the Controlled CPS Vector.

export_beyond_trust: This 1-bit field indicates that content may be exported to an untrusted space.

disable_analogue_sd_export: This 1-bit field indicates that export of standard definition analogue content is prohibited.

disable_analogue_sd_consumption: This 1-bit field indicates that output of standard definition analogue content for consumption is prohibited.

disable_analogue_hd_export: This 1-bit field indicates that that export of high definition analogue content is prohibited.

disable_analogue_hd_consumption: This 1-bit field indicates that that output of high definition analogue content for consumption is prohibited.

image_constraint: This 1-bit field indicates that high definition images shall be rendered at lower resolutions both for analogue outputs and exports to an untrusted space.

view_window_start: This 40-bit field indicates the absolute start time from which the view window is enabled, or opened.

view_window_end: This 40-bit field indicates the absolute end time until which the view window is enabled, or opened.

view_period_from_first_playback: This 16-bit field indicates the period during which Consumption or Export is allowed after the first Consumption or Export of that CPCM Content has been initiated.

simultaneous_view_count: This 8-bit field indicates the total number of concurrent consumption points and export points that can be used for the live/direct consumption and export of a content item.

remote_access_date: This 40-bit field indicates the absolute time after which remote access to the content item is enabled.

cps_vector_count: This field indicates the number of CPS vectors.

C_and_R_regime_mask: This field identifies under which of the 8 C&R regimes the associated CPS vector is covered.

cps_vector_length: This field indicates the length in bytes of the CPS vector.

cps_vector: Each bit of this vector indicates that content may be exported or output to a given controlled copy protection system. The mapping of cps_vector bits to controlled copy protection systems is defined by C&R regimes and beyond the scope of the present document.

5.4.5 CPCM delivery signalling

CPCM delivery signalling is CPCM Content metadata that signals the Authorized Usage to be applied to CPCM Content during Acquisition. Carriage of CPCM delivery signalling is defined in detail in TS 102 825-9 [i.3]. Table 12 shows the structure of CPCM delivery signalling.

Table 12: Version 1 CPCM delivery signalling

Syntax	Number of bits	Identifier
cpcm_v1_delivery_signalling(){		
copy_control	3	uimsbf
do_not_cpcm_scramble	1	bslbf
viewable	1	bslbf
view_window_activated	1	bslbf
view_period_activated	1	bslbf
simultaneous_view_count_activated	1	bslbf
move_local	1	bslbf
view_local	1	bslbf
move_and_copy_propagation_information	2	uimsbf
view_propagation_information	2	uimsbf
remote_access_delay_flag	1	bslbf
remote_access_date_flag	1	bslbf
remote_access_record_flag	1	bslbf
export_controlled_cps	1	bslbf
export_beyond_trust	1	bslbf
disable_analogue_sd_export	1	bslbf
disable_analogue_sd_consumption	1	bslbf
disable_analogue_hd_export	1	bslbf
disable_analogue_hd_consumption	1	bslbf
image_constraint	1	bslbf
if (view_window_activated == 1){		
view_window_start	40	bslbf
view_window_end	40	bslbf
}		
if (view_period_activated == 1){		
view_period_from_first_playback	16	uimsbf
}		
if (simultaneous_view_count_activated == 1){		
simultaneous_view_count	8	uimsbf
}		
if (remote_access_delay_flag == 1){		
remote_access_delay	16	uimsbf
}		
if (remote_access_date_flag == 1){		
remote_access_date	40	bslbf
}		
if (export_controlled_cps == 1){		
cps_vector_count	8	
for (i=0; i<cps_vector_count; i++){		
C_and_R_regime_mask		
cps_vector_length	16	uimsbf
for (j=0; j<N; j++){		

Syntax	Number of bits	Identifier
cps_vector_byte	8	bslbf
}		
}		
}		
}		

Semantics for the Version 1 CPCM delivery signalling:

All fields have the same meaning as defined in TS 102 825-3 [i.11], with the exceptions of the following fields:

remote_access_delay_flag: This 1-bit field indicates that remote access is permitted after delay indicated by the remote_access_delay field.

remote_access_date_flag: This 1-bit field indicates that remote access is permitted after the date indicated by the remote_access_date field.

remote_access_delay: This 16-bit field indicates the period of time in 15 minute increments relative to the acquisition of the content item after which remote access is enabled.

remote_access_date: This 40-bit field indicates the absolute time in Universal Time, Co-ordinated (UTC) and Modified Julian Date (MJD) (see annex C) after which remote access to the content item is enabled. This field is coded as 16 bits giving the 16 LSBs of MJD followed by 24 bits coded as 6 digits in 4-bit Binary Coded Decimal (BCD). If the start time is undefined (e.g. for an event in a NVOD reference service) all bits of the field are set to "1".

EXAMPLE: 93/10/13 12:45:00 is coded as "0xC079124500".

5.4.6 CPCM status information

The CPCM_status_information structure is used to communicate the status of a CPCM Instance as a result of a request from another CPCM Instance. It is a structure containing a collection of static information duplicated from the CPCM Instance Certificate (CIC), defined in clause 5.4.2, and the status of dynamic ADM-related variables, defined from clause 5.4.17 onwards. Table 13 shows the format of CPCM status information.

Table 13: CPCM status Information

Syntax	Bits	Identifier
CPCM_status_information(){		
CPCM_version	8	uimsbf
C_and_R_regime_mask	8	bslbf
reserved	3	bslbf
content_handling_capability	5	bslbf
AD_aware	1	bslbf
ADM_capable	1	bslbf
ADM_LM_capable	1	bslbf
ADM_DC_capable	1	bslbf
ADSE_countable	1	bslbf
LSA_capable	1	bslbf
absolute_time_capable	1	bslbf
geographic_aware	1	bslbf
AD_identifier	72	ADID
}		

Semantics for CPCM_status_information:

CPCM_version: CPCM System version supported by the CPCM Instance. CPCM Instances supporting the present document shall set CPCM_version to 0x01. CPCM_version may increment with later revisions of the present document.

C_and_R_regime_mask: Identifies which of the 8 C&R regimes is, or are supported.

content_handling_capability: Bit mask for the CPCM Functional Entities implemented in the CPCM Instance, with 1 bit for each of APECS, in that order.

AD_aware: Whether or not the CPCM Instance is able to enforce AD-related USI.

ADM_capable: Whether or not the CPCM Instance is capable of performing ADM functionality.

ADM_LM_capable: Whether or not the CPCM Instance can act as an ADM Local Master.

ADM_DC_capable: Whether or not the CPCM Instance can act as an ADM Domain Controller.

ADSE_countable: Whether or not the CPCM Instance shall affect ADM ADSE counts.

LSA_capable: Whether or not the CPCM Instance is capable of scrambling and/or descrambling CPCM Content (using LSA).

absolute_time_capable: Whether or not the CPCM Instance is capable of maintaining secure absolute time.

geographic_aware: Whether or not the CPCM Instance is capable of determining its geographic location by its own means.

AD_identifier: The ADID to which the CPCM Instance belongs. This shall be set to 0 if it does not belong to any AD.

The **reserved** bits shall be set to zero. Future versions of the CPCM specification may utilize the reserved bits.

5.4.7 CPCM Content handling operation

The CPCM_content_handling_operation structure is used to signal the application of one or more of the five CPCM content handling operations (APECS), as defined in the CPCM Reference Model (TS 102 825-2 [i.5]).

Table 14 shows the structure of CPCM Content Handling Operation.

Table 14: CPCM Content handling operation

Syntax	Bits	Identifier
CPCM_content_handling_operation(){		
reserved	3	bslbf
acquisition	1	bslbf
processing	1	bslbf
export	1	bslbf
consumption	1	bslbf
storage	1	bslbf
}		

Semantics for CPCM_content_handling_operation:

acquisition: This bit shall be set to 0 if no Acquisition operation is indicated and shall be set to 1 when an Acquisition operation is indicated.

processing: This bit shall be set to 0 if no Processing operation is indicated and shall be set to 1 when a Processing operation is indicated.

export: This bit shall be set to 0 if no Export operation is indicated and shall be set to 1 when an Export operation is indicated.

consumption: This bit shall be set to 0 if no Consumption operation is indicated and shall be set to 1 when a Consumption operation is indicated.

storage: This bit shall be set to 0 if no Storage operation is indicated and shall be set to 1 when a Storage operation is indicated.

5.4.8 CPCM Revocation List

Table 15 shows the format of the CPCM Revocation List.

Table 15: Revocation List format

Syntax	Bits	Identifier
CPCM_revocation_list(){		
CPCM_version	8	uimsbf
RL_index	32	uimsbf
C_and_R_regime_mask	8	bslbf
certificate_generation_index	8	uimsbf
certificate_generation_RL_index_change	32	uimsbf
revoked_certificate_count	32	uimsbf
revoked_AD_count	32	uimsbf
for(i=0; i<revoked_certificate_count; i++){		
index_of_first_revocation	32	uimsbf
revoked_certificate_identifier	64	uimsbf
}		
for(i=0; i<revoked_AD_count; i++) {		
index_of_first_revocation	32	uimsbf
revoked_ADS_digest	160	bslbf
}		
signature	2 048	uimsbf
}		

The total size of the CPCM_revocation_list record is N + (12*revoked_certificate_count) + (24*revoked_AD_count).

Semantics for CPCM_revocation_list:

CPCM_version: CPCM System version supported by the CPCM Revocation List. CPCM Revocation Lists supporting the present document shall set CPCM_version to 0x01. CPCM_version may increment with later revisions of the present document.

RL_index: A number incremented as each new Revocation List is issued.

C_and_R_regime_mask: Identifies the C&R regime(s) that issued the CPCM Revocation List. At least one bit shall be set.

certificate_generation_index: Minimal Generation Index a certificate shall have. Certificates with lower index shall be considered to be revoked.

certificate_generation_RL_index_change: Index of the Revocation List when the Certificate Generation index last changed.

revoked_certificate_count: Number of Revoked Certificates contained in this CPCM revocation list.

revoked_AD_count: Number of Revoked ADs contained in this CPCM revocation list.

index_of_first_revocation: Revocation List Index when the respective Certificate or AD was first revoked.

revoked_certificate_identifier: The CPCM_instance_certificate_id of the revoked Certificate.

revoked_ADS_digest: AD Secret digest of the revoked AD.

signature: Signature of the CPCM Revocation List. See (TS 102 825-5 [i.8]) for the algorithm applied to create this signature.

5.4.9 CPCM Auxiliary Data

The CPCM_auxiliary_data structure is used to carry CPCM Auxiliary Data as shown in table 16. The CPCM Auxiliary Data structure is securely bound to the CL but transported independently as specified in table 16.

Table 16: CPCM Auxiliary Data structure

Syntax	Bits	Identifier
CPCM_auxiliary_data(){		
element_counter	8	uimsbf
for (i=0; i<element_counter; i++){		
CPCM_auxiliary_data_element_identifier	8	bslbf
[CPCM_auxiliary_data_element_length]	16	uimsbf
CPCM_auxiliary_data_element		
}		
}		

Semantics for CPCM_auxiliary_data:

CPCM_auxiliary_data_element_counter: The total number of CPCM auxiliary data elements.

CPCM_auxiliary_data_element_identifier: The identifier of the element carried, as defined in table 17.

CPCM_auxilary_data_element_length: This field indicates the length in bytes of the element, but only for variable-length elements, as identified in table 17. 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.

CPCM_auxiliary_data_element: A CPCM Auxiliary Data element.

Table 17: CPCM auxiliary data element identifiers

Element Identifier	Element Name	Bits	Identifier	Description
0	original_full_USI	variable	bslbf	Original full USI as provided by the delivery system within the Acquisition Point. See TS 102 825-3 [i.11].
1	original_FTA_control	8	bslbf	Original signalling associated with Free-To-Air content. See EN 300 468 [1].
2	CPCM_geographic_area	variable	bslbf	Geographic location information for Content marked MGAD or VGAD. See clause 5.4.12.
3	other_CPS_export_data	variable	bslbf	Proprietary data aimed at the other CPS to which the Content may be Exported. Other CPS can be Trusted, Controlled, or an Analogue Output. See TS 102 825-10 [i.10].
4 to 5	reserved			
6	CPCM_rights_issuer_URL	variable	bslbf	URL where additional Authorized Usage rights for the associated CPCM Content Item can be obtained.
7	CLC_private_data	variable	bslbf	Private data associated with the proprietary system from which the CPCM Content was Acquired.
8	CPCM_extension_data	variable	bslbf	Data destined for a CPCM extension.
9 to 255	reserved	variable	bslbf	Any newly defined element shall be accompanied by its corresponding CPCM_auxiliary_data_element_length field.

5.4.10 CPCM geographic location information

The CPCM geographic information structure carries the geographic area to which MGAD or VGAD restriction applies. Its structure is shown in table 18.

Table 18: CPCM geographic location information structure

Syntax	Bits	Identifier
CPCM_geographic_location_information(){		
geo_location_format	8	CPCM_geo_location_format_code
[geo_location_element_length]	8	uimsbf
geo_location()		see Table 19
}		

Semantics for CPCM_geographic_location_information:

geo_location_format: The geographic location format code, see table 19.

geo_location_element_length: This field indicates the length in bytes of the geo_location element, but only for variable-length elements, as identified in tTable 19. 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.

geo_location(): The actual geographic location, see table 19.

Table 19: CPCM geographic location format identifiers

CPCM Geographic Location Format Identifier	CPCM Geographic Location Format	CPCM Geographic Location Format Size
0x00	reserved	
0x01	country code	2 bytes
0x02	country code (excluded)	2 bytes
0x03	2D absolute position	6 bytes
0x04	URL pointing to location from which geographical area can be fetched	variable
0x05-0xFF	reserved for future extension	

Semantics for CPCM_geo_location_format:

country code: This is a 2-byte alphanumerical country code as defined in ISO 3166-1 [3]. Additionally, the code 'AA' defines the whole world.

country code (excluded): This is a 2-byte alphanumerical country code as defined in ISO 3166-1 [3]. This is used in Auxiliary Data to exclude remote access to the CPCM content from within the indicated territory or territories.

2D absolute position: This is a coding of an absolute geographical position in terms of latitude and longitude. It is not used to signal any Authorized Usage for CPCM Content. It is coded as follows:

```
Latitude (24-bit bslbf)
{±ddddddd:mmmmmm:ssssss.sss}
    {msb → 0 = +; 1 = -}
    {8 bit "degree field" → 0 - 180; 181-255 reserved}
    {6 bit "minute field" → 0 - 59; 60-63 reserved}
    {6 bit "second field" → 0 - 59; 60-63 reserved}
    {3 bit "second fraction field" →  integer number of 1/8 seconds}
Longitude (24-bit bslbf)
{±ddddddd:mmmmmm:ssssss.sss}
```

URL pointing to location: This is a URL that points to a web page from which the CPCM_geographic_area structure (see clause 5.4.12) can be read.

5.4.11 CPCM geographic location formats list

The CPCM_geographic_location_formats_list structure carries a list of geographic location format identifiers, as shown in table 20.

Table 20: CPCM geographic format codes list structure

Syntax	Bits	Identifier
CPCM_geographic_format_codes_list(){		
format_code_count	8	uimsbf
for (i=0; i<format_code_count; i++){		
CPCM_geographic_location_format_identifier	8	uimsbf
}		
}		

Semantics for CPCM_geographic_format_codes_list:

format_code_count: The number of CPCM geographic format codes contained in the list.

CPCM_geographic_location_format_code: An identifier from table 19.

5.4.12 CPCM geographic area format

The CPCM_geographic_area structure carries a list of geographic location , as shown in table 21.

Table 21: CPCM geographic area structure

Syntax	Bits	Identifier
CPCM_geographic_area(){		
location_count	8	uimsbf
for (i=0; i<location_count; i++){		
CPCM_geographic_location_information	8	uimsbf
}		
}		

Semantics for CPCM_geographic_area:

location_count: The number of CPCM geographic locations in the list.

CPCM_geographic_location_information: A location information as specified in clause 5.4.10.

5.4.13 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. Its structure is shown in table 22. CPCM Instances that do not implement the indicated other CPS shall ignore the associated private data.

Table 22: Other CPS export data structure

Syntax	Bits	Identifier
other_CPS_export_data(){		
CP_system_id	16	uimsbf
other_CPS_export_data_length	16	uimsbf
for (i=0; i<other_CPS_export_data_length; i++){		
char	8	bslbf
}		

Semantics for other_CPS_export_data:

CP_system_id: This 16-bit field identifies the other CPS to which the private data applies. Allocations of the value of this field are defined in ETR 162 [4].

other_CPS_export_data _length: The length of the other_CPS_export_data, as the number of bytes, maximum 65,536, of data following the length field.

char: Other CPS export data byte.

5.4.14 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. Its structure is shown in table 23.

Table 23: CPCM rights issuer URL structure

Syntax	Bits	Identifier
CPCM_rights_issuer_URL(){		
URL_length	8	uimsbf
for (i=0; i<URL_length; i++){		
char	8	bslbf
}		

Semantics for CPCM_rights_issuer_URL:

URL_length: The length of the URL, as the number of bytes contained in the URL, maximum 255, of UTF-8 coded text following the length field.

char: UTF-8 coded character of the URL.

5.4.15 CLC private data

The CLC_private_data structure carries data that is specific to a proprietary system, for example the CA system from which the CPCM Content Item was Acquired. Its structure is shown in table 24. CPCM Instances that do not implement the indicated CA system shall ignore the associated private data.

Table 24: CLC private data structure

Syntax	Bits	Identifier
CLC_private_data(){		
CA_system_id	16	uimsbf
CLC_private_data_length	16	uimsbf
for (i=0; i<CLC_private_data_length; i++){		
char	8	bslbf
}		

Semantics for CLC_private_data:

CA_system_id: This 16-bit field identifies the CA system to which the private data applies. Allocations of the value of this field are found in ETR 162 [4].

CLC_private_data _length: The length of the CLC_private_data, as the number of bytes, maximum 65,536, of data following the length field.

char: CLC private data byte.

5.4.16 CPCM extension private data

The CPCM_extension_data structure carries data that is specific to a particular CPCM extension.. Its structure is shown in table 25. CPCM Instances that do not implement the indicated CPCM extension shall ignore the associated private data.

Table 25: CPCM extension private data structure

Syntax	Bits	Identifier
CPCM extension_data(){		
CPCM_extension_id	16	uimsbf
CPCM_extension_data_length	16	uimsbf
for (i=0; i<CPCM_extension_data_length; i++){		
char	8	bslbf
}		

Semantics for CPCM_extension_data:

CPCM_extension_id: This 16-bit field identifies the CPCM extension to which the private data applies. Allocations of the value of this field are yet to be defined.

CPCM_extension_data _length: The length of the CPCM_extension_data, as the number of bytes, maximum 65,536, of data following the length field.

char: CPCM_extension data byte.

5.4.17 AD name

The AD_name structure is used to store the name of an Authorized Domain (AD) as shown in table 26.

Table 26: AD name structure

Syntax	Bits	Identifier
AD_name(){		
reserved	3	bslbf
AD_name_length	5	uimsbf
for (i=0; i<AD_name_length; i++){		
char	8	bslbf
}		
}		

Semantics for AD_name:

The **reserved** bits shall be set to zero. Future versions of the CPCM specification may utilize the reserved bits.

AD_name_length: The length of the AD name, as the number of bytes, maximum 31, of UTF-8 coded text following the length field.

char: UTF-8 coded character of the AD name.

5.4.18 AD capability

The AD_capability structure carries the replication of the five AD and ADM capability fields of the CPCM Instance Certificate, in the same order as in the certificate, as shown in table 27.

Table 27: AD capability structure

Syntax	Bits	Identifier
AD_capability(){		
reserved	3	bslbf
AD_aware	1	bslbf
ADM_capable	1	bslbf
ADM_LM_capable	1	bslbf
ADM_DC_capable	1	bslbf
ADSE_countable	1	bslbf
}		

Semantics for AD_capability:

The **reserved** bits shall be set to zero. Future versions of the CPCM specification may utilize the reserved bits.

AD_aware: Whether or not the CPCM Instance is able to enforce AD-related USI.

ADM_capable: Whether or not the CPCM Instance is capable of performing ADM functionality.

ADM_LM_capable: Whether or not the CPCM Instance can act as an ADM Local Master.

ADM_DC_capable: Whether or not the CPCM Instance can act as an ADM Domain Controller.

ADSE_countable: Whether or not the CPCM Instance shall affect ADM ADSE counts.

5.4.19 ADSE values

The ADSE_values structure, shown in table 28, carries the current state of the ADSE variables associated with an AD. Each of these variables is defined in the ADM specification (TS 102 825-7 [i.9]).

Table 28: ADSE values structure

Syntax	Bits	Identifier
ADSE_values(){		
total_count	16	uimsbf
total_ceiling	16	uimsbf
remote_count	16	uimsbf
remote_ceiling	16	uimsbf
local_count	16	uimsbf
local_ceiling	16	uimsbf
DC_split_count	16	uimsbf
DC_split_ceiling	16	uimsbf
DC_remote_count	16	uimsbf
DC_remote_ceiling	16	uimsbf
}		

Semantics for ADSE_values:

total_count: The current number of CICF in the Authorized Domain.

total_ceiling: The maximum number of CICF allowed in the Authorized Domain.

remote-count: The current number of CICF in the AD that Joined Remotely to the Domain Controller that authorized the AD Join.

remote_ceiling: The maximum number of CICF that may Remotely Join the AD.

local_count: The current number of CICF in the AD that have Joined Locally to the Domain Controller that authorized the AD Join.

local_ceiling: The maximum number of CICF that may Locally Join the AD without running the Quorum test.

DC_split_count: The current count of Domain Controller Splits.

DC_split_ceiling: The maximum number of DC Splits that may occur.

DC_remote_count: The current count of Remote Transfer or Remote Splits of the Domain Controller.

DC_remote_ceiling: The maximum number of Remote Transfer or Remote Splits of the Domain Controller.

5.4.20 DC ADSE value list

The DC_ADSE_values_list structure carries a list of Domain Controllers and their respective ADSE values structures, as shown in table 29.

Table 29: DC ADSE Values list structure

Syntax	Bits	Identifier
DC_ADSE_values(){		
DC_count	8	uimsbf
for (i=0; i<DC_count; i++){		
DC_id	64	CPCM_instance_certificate_id
DC_ADSE_values()	160	ADSE_values
}		
}		

Semantics for DC_ADSE_values:

DC_count: The number of DC_ADSE_values structures contained in the list.

DC_id: The CPCM Instance Certificate identifier of the Domain Controller associated with the ADSE values.

DC_ADSE_values: ADSE values (see clause 5.4.19).

5.4.21 DC Local identifiers list

This structure is used to carry a list of Domain Controller (DC) local identifiers, as shown in table 30.

Table 30: DC Local identifiers list information element structure

Syntax	Bits	Identifier
DC_local_id_list(){		
local_id_count	8	uimsbf
for (i=0; i<local_id_count; i++){		
DC_local_id	8	uimsbf
}		
}		

Semantics for DC_local_id_list:

local_id_count: The number of DC local identifiers in the list.

DC_local_id: An 8-bit number used as an identifier of a Domain Controller within the Authorized Domain and used in the AD internal record (see clause 5.4.22).

5.4.22 AD internal record

The AD_internal_record structure is used to carry an AD internal record.

Table 31: Domain Internal Record Information Element Structure

Syntax	Bits	Identifier
AD_internal_record(){		
ADID	72	bslbf
AD_controller_id	64	CPCM_instance_certificate_id
AD_controller_local_id()	8	bslbf
internal_record_index	16	uimsbf
ADSE_values()	160	ADSE_values
current_DC_local_ids()		DC_local_id_list
merged_DC_local_ids()		DC_local_id_list
change_date	40	CPCM_date_time
}		

Semantics for AD_internal_record:

ADID: The ADID of the Domain to which the AD internal record applies.

AD_controller_id: The CPCM Instance identifier of the Domain Controller that created the AD internal record.

AD_controller_local_id: The local identifier of the Domain Controller that created the AD internal record.

internal_record_index: The index of the AD internal record.

ADSE_values: The ADSE values associated with the Domain Controller that issued the AD internal record.

current_DC_local_ids: The list of local identifiers of the current Domain Controllers.

merged_DC_local_ids: The list of local identifiers of Domain Controllers that have been Merged.

change_date: The date when the AD internal record created. This field being optional, 0 means the date was not given.

5.4.23 AD internal record list

The AD_internal_record_list structure is used to carry a list of Domain internal records.

Table 32: Merged DC Local identifiers list information element structure

Syntax	Bits	Identifier
AD_internal_record_list(){		
internal_record_count	8	uimsbf
for (i=0; i< count_internal_record; i++){		
internal_record()		AD_internal_record
}		
}		

Semantics for AD_internal_record_list:

internal_record_count: The number of AD internal records in the list.

internal_record: An AD internal record (see clause 5.4.22).

5.4.24 CPCM extension data element

This structure allows data to be sent to a CPCM Extension.

Table 33: CPCM extension data element structure

Syntax	Bits	Identifier
CPCM_extension_element(){		
CPCM_extension_identifier	16	bslbf
CPCM_extension_length	16	uimsbf
CPCM_extension_data		
}		

Semantics for CPCM_extension_element:

CPCM_extension_identifier: The identifier of the CPCM Extension. These are not specified by the present specification. CPCM Extension identifiers may be allocated by a relevant registration authority.

CPCM_extension_length: The length of the CPCM Extension data.

CPCM_extension_data: The CPCM Extension data.

5.4.25 Private element structure

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

Table 34: Private data element structure

Syntax	Bits	Identifier
private_element(){		
private_element_tag	16	uimsbf
private_element_data_length	16	uimsbf
private_element_data		
}		

Semantics for private_element:

private_element_tag: The identifier of the private data. It will be attributed by a relevant registration authority.

private_element_data_length: The length of the private data.

private_element_data: The private data to be communicated.

6 CPCM protocols

6.1 General

This clause provides a normative, generic specification of the protocols needed to implement the CPCM System.

Clause 6.2 defines the generic CPCM protocol message framework, which includes a facility for protocol transactions.

The set of CPCM protocols is divided into several categories for convenience, according to the constituent elements of the CPCM Instance, namely Security Toolbox (SEC), CPCM System and Content Management (SYS), Authorized Domain Management (ADM). There are also some general message categories, namely a bridging protocol, generic frameworks for extension messages and private messages, and a generic CPCM protocol error message.

Each of the protocol message categories is specified from clause 6.3 onwards. Each CPCM protocol step is defined using two distinct methodologies, firstly as a more illustrative C-type function call, and secondly in the form of a CPCM protocol message structure. The C-type function call representation is in effect an abstraction of the respective protocol message. Conversely, the protocol message in effect simply encapsulates the C-type function call inside a dedicated message header that contains a message code and a control field.

Any deployment of the CPCM System must apply a home network ecosystem adaptation layer (defined in TS 102 825-9 [i.3]), and/or a CPCM application-specific device interface (defined in TS 102 825-9 [i.3]), where it is specified how the generic CPCM protocols are mapped to work on the respective ecosystem or interface.

When specifying a CPCM System adaptation for a particular HN ecosystem or application-specific physical interface it may be advantageous to apply one methodology rather than the other when specifying the mapping of the CPCM protocols to that entity, i.e. to either specify the CPCM protocol calls within that ecosystem, or to utilize a generic messaging mechanism within that ecosystem to encapsulate CPCM protocol messages.

6.2 CPCM protocol message framework

6.2.1 General

This clause defines the generic CPCM protocol messaging framework upon which all message-based implementations of the CPCM protocols are based.

A CPCM protocol corresponds to a function, or transaction, to be performed within the CPCM System. A CPCM protocol may consist of one or more CPCM protocol steps, which correspond to one or more function calls within that function, or transaction.

A CPCM protocol (function, or transaction) may consist of any number of CPCM protocol steps.

Each CPCM protocol step is implemented as a discrete CPCM protocol message. The generic CPCM protocol message is defined in clause 6.2.2.

If the receiving CPCM Instance is not able to process the CPCM protocol message sent to it by another CPCM Instance then it shall respond to that CPCM Instance with an appropriate CPCM protocol error message. The generic CPCM protocol error message is defined in clause 6.3.

If the receiving CPCM Instance is able to process the CPCM protocol message sent to it by another CPCM Instance then it shall not respond with any CPCM protocol error message (i.e. there is no CPCM protocol error message to indicate "success" or to acknowledge the receipt of a CPCM protocol message). In this case the receiving CPCM Instance proceeds by performing the requested function call and then by progressing to the next step of the CPCM protocol, by sending the appropriate CPCM protocol message and associated information to the first CPCM Instance.

A CPCM protocol may be terminated either by a protocol-specific "success" or "acknowledge" message, sent upon successful completion of the CPCM protocol, or by a CPCM protocol error message. If a CPCM protocol error message is sent then the next CPCM protocol step shall not be undertaken by the CPCM Instance that receives such a CPCM protocol error message, and the CPCM protocol that had been initiated is backed out by both CPCM Instances.

The generic framework is depicted in figure 3, which shows two CPCM Instances, "A" and "B", performing a CPCM protocol that consists of four steps, whereby CPCM Instance A initiates the CPCM protocol with CPCM Instance B. In this diagram all possible CPCM protocol error messages are illustrated, but in practice if any one of these occurs then the next CPCM protocol step shall not be undertaken.

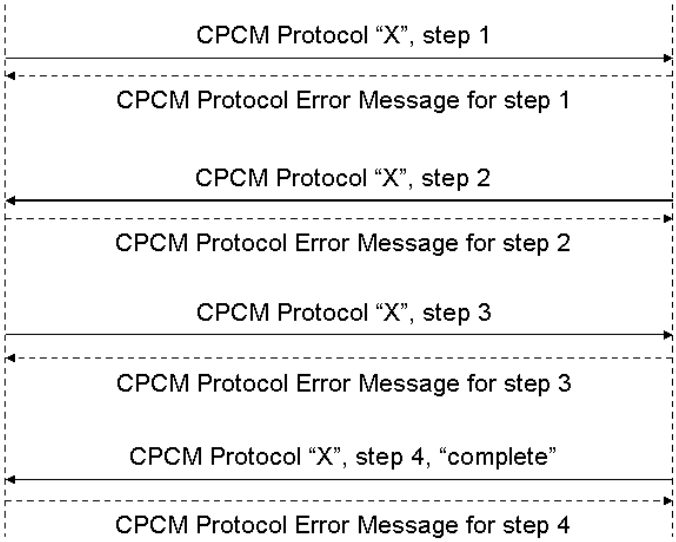

Figure 3: Generic CPCM protocol flow

CPCM protocols shall be performed sequentially between two CPCM Instances, thus a CPCM Instance shall not initiate more than one CPCM protocol (function, or transaction) of the same CPCM protocol message type with the same CPCM Instance at any time.

The CPCM System foresees a general protocol timeout period, which is said to have occurred if an expected protocol step message is not received within a certain time since the previous protocol step. This time duration is not set by the present document, rather it is left to be set by the implementation or by the C&R regime. It is expected to be of the order of a few seconds.

The CPCM System also requires a broadcast mode for messages within the realms of a home network ecosystem. Figure 4 shows the generic representation of a CPCM protocol initiated with a broadcast message (by nominal CPCM Instance A) and when other CPCM Instances respond also with broadcast messages (here, nominal CPCM Instances B and C).

The CPCM System foresees a broadcast response timeout period within which all such responses shall be registered, but the present document does not stipulate the value for this timeout. This shall be set by the implementation or by the C&R regime.

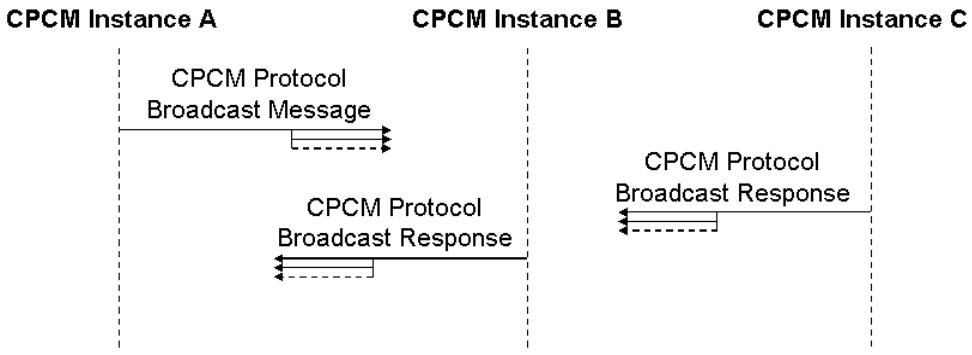

Figure 4: Generic CPCM broadcast protocol, broadcast responses

Figure 5 shows a generic CPCM protocol that is initiated with a broadcast message, whereby unicast responses shall be returned. The same timeout value also applies for this case.

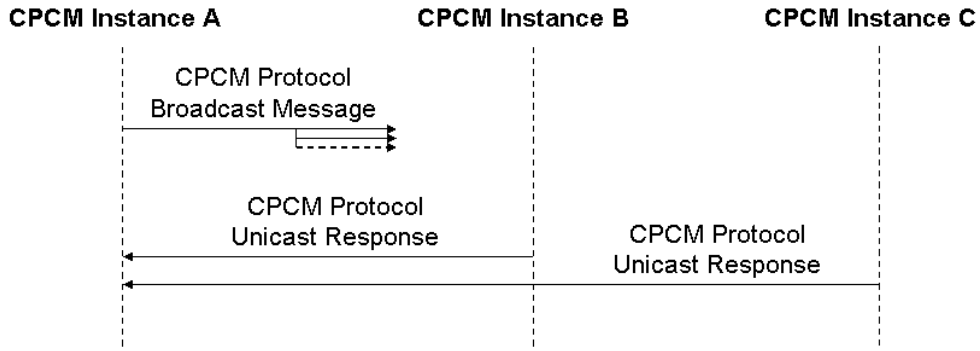

Figure 5: Generic CPCM Broadcast Protocol, Unicast responses

6.2.2 CPCM protocol message

6.2.2.1 General

CPCM protocol message is the generic structure used to exchange information between two CPCM Instances. This generic structure is used to define the CPCM protocols in the form of dedicated messages corresponding to each CPCM protocol step. The generic structure of CPCM protocol message is shown in table 35.

The default basic structure with a variable-length message payload, whereby each protocol message defines a fixed-length structure to carry the data needed for that message.

Some CPCM protocol messages contain conditional elements that are carried only in certain circumstances specific to that protocol.

Some CPCM protocol messages may contain optional elements that are carried according to implementation choice.

Table 35: CPCM protocol message

Syntax	Bits	Identifier
CPCM_protocol_message(){		
CPCM_protocol_message_type()	16	uimsbf
control_field	8	uimsbf
message_payload_length	16	uimsbf
for (i=0;i<message_payload_length;i++){		
message_payload_byte	8	bslbf
}		
[conditional elements()]		
[optional elements()]		
}		

Semantics for CPCM_protocol_message:

CPCM_protocol_message_type: This is the identifier of the protocol message, defined in clause 6.2.2.5.

control_field: This field indicates certain properties of the protocol message as specified in clause 6.2.2.6.

message_payload_length: This field indicates the number of message_payload_bytes of the message plus the length of the message signature, if the message is signed.

message_payload_byte: CPCM messages may contain payload bytes, as specified for each CPCM protocol message type in clause 6.3 onwards.

conditional_elements: See clause 6.2.2.2.

optional_elements: See clause 6.2.2.3.

6.2.2.2 Conditional elements

The conditional_elements structure is used to carry additional mandatory data fields within CPCM protocol messages as defined for each individual CPCM protocol message where they are needed.

Table 36: Conditional Elements Structure

Syntax	Bits	Identifier
conditional_elements(){		
conditional_element_count	8	uimsbf
for (i=0;i<conditional_element_count;i++){		
element_identifier	8	uimsbf
[element_length]	16	uimsbf
conditional_element		
}		
}		

Semantics for conditional_elements:

element_identifier: The identifier of the element carried, as defined in table 38.

element_length: This field indicates the length in bytes of the element, but only for variable-length elements, as identified in table 38. This field is omitted for fixed-length elements. element_length is mandatory for variable length elements and for any new elements that would be defined in a new version of the present document.

6.2.2.3 Optional elements

In general, the format of any given field or structure shall remain unchanged for all future versions of the present document. Where it is necessary to carry additional information, new fields or structures shall be defined for this purpose and included in the conditional elements structure where necessary. If an Instance receives an Element Type that it does not understand, e.g. for a later version of the specification then the Element shall be ignored.

The optional_elements structure is used to carry additional optional data fields within CPCM protocol messages.

Table 37: Optional Elements Structure

Syntax	Bits	Identifier
optional_elements(){		
optional_element_count	8	uimsbf
For (i=0;i<optional_element_count;i++){		
element_identifier	8	uimsbf
[element_length]	16	uimsbf
optional_element		
}		
}		

Semantics for optional_elements:

element_identifier: The identifier of the element carried, as defined in table 38.

element_length: This field indicates the length in bytes of the element, but only for variable-length elements, as identified in table 38. This field is omitted for fixed-length elements. element_length is mandatory for variable length elements and for any new elements that would be defined in a new version of the present document.

6.2.2.4 Element identifiers

Whenever it is necessary to carry one or more elements in either the conditional, or optional elements structures, the carried elements shall be identified by the element identifiers as listed in table 38.

Table 38: CPCM protocol element identifiers

Element Identifier	Element Name	Bits	Identifier	Description
1	ADID	72	bslbf	The ADID shall be used to carry the globally unique, permanent identifier for the Domain.
2	originating_instance_id	64	bslbf	The unique permanent identifier of the CPCM Instance originating the operation.
3	destination_instance_id	64		The unique permanent identifier for the destination CPCM Instance. Note that this may be different from the initial receiving Instance, in which case it is required to relay the request to the final destination.
4	delegating_instance_id	64	bslbf	See clause 6.6.2.5.
5	AD_name()	<248		See clause 5.4.17.
6	original_message_type	16	bslbf	This element contains the 16 bit message type of the originating message; it is only used in error messages.
7	AD_capability()	8	bslbf	See clause 5.4.18.
8	ADSE_values()	160	bslbf	See clause 5.4.19.
9	domain_controller_id	64	CPCM_instance_id	The globally unique permanent identifier for a current Domain Controller for this Domain.
10	DC_ADSE_values()	variable		See clause 5.4.20.
11	internal_record_index	16	uimsbf	The index of a Domain Internal Record.
12	change_date	40	uimsbf	A Date/Time value, either to indicate the last update time for a set of data, or the current time as known to the originating Instance.
13	LM_capability	8	uimsbf	See clause 6.6.2.6.
14	CPCM_version	8	uimsbf	
15	error_code	8	uimsbf	See clause 6.3.2.
16	challenge	128	bslbf	A random value used in cryptographic protocols (Proximity Tests, Quorum Tests).
17	ADM_status	8	uimsbf	See clause 6.6.2.2.
18	AD_condition	8	uimsbf	See clause 6.6.2.3.
19	AD_protocol	8	uimsbf	See clause 6.6.2.4.
20	domain_controller_local_id	8	bslbf	The local identifier of a Domain Controller.
21	current_DC_local_ids()	variable		See clause 5.4.21.
22	merged_DC_local_ids()	variable		See clause 5.4.21.
23	internal_record()	variable		See clause 5.4.22.
24	internal_record_list()	variable		See clause 5.4.23.
25	CPCM_AD_secret	128		See TS 102 825-7 [i.9].
26	CPCM_extension_element ()	variable		See clause 5.4.24.
27	private_element()	variable		See clause 5.4.25.

6.2.2.5 CPCM protocol message type

The CPCM protocol message type is a 16-bit code that identifies the protocol call of the message packet in which it appears.

The CPCM protocol message type is structured such that the first 8 bits denote the message category and the last 8 bits the protocol call within that category, as shown in table 39.

Table 39: CPCM protocol message type

Syntax	Bits	Identifier
CPCM_protocol_message_type(){		
message_type_category_id	8	uimsbf
message_code	8	uimsbf
}		

Semantics for CPCM_protocol_message_type:

message_type_category_id: The category of the message, as defined in table 40.

message_code: The message code of the message, as defined in the relevant CPCM protocol section.

The division of messages into categories and the pre-allocation of message type ranges for each category enable easier revision of protocol calls within each category in possible future revisions of the present document.

Table 40 lists the CPCM protocol message type categories and their identifiers.

Table 40: CPCM protocol message type categories

Message type category identifier	Message type category
0x00	Reserved
0x01	CPCM Security Control (SEC)
0x02	SEC messages defined in adaptation layers
0x03 to 0x3F	Reserved for SEC extensions
0x40	CPCM System / Content management (SYS)
0x41	SYS messages defined in adaptation layers
0x42 to 0x5F	Reserved for CPCM System extensions
0x60	Authorized Domain Management (ADM)
0x61	ADM relay messages
0x62 to 0x7F	Reserved for ADM extensions
0x80 to 0xEF	Reserved
0xF0	CPCM Extension messages
0xF1 to 0xFD	Reserved
0xFE	Private messages
0xFF	General CPCM protocol messages

The CPCM protocol message codes for each message type category are defined in the corresponding clause.

6.2.2.6 Control field

The control_field structure contains flags that indicate which CPCM security tools have been applied to protect the message. Table 41 shows the structure of control_field.

Table 41: Control field

Syntax	Bits	Identifier
`control_field(){`		
`  reserved`	4	bslbf
`  payload_signed_SAC`	1	bslbf
`  payload_signed_ADS`	1	bslbf
`  payload_encrypted_SAC`	1	bslbf
`  payload_encrypted_ADS`	1	bslbf
`}`		

Semantics for control_field:

The **reserved** bits shall be set to zero. Future versions of the CPCM specification may utilize the reserved bits.

payload_signed_SAC: This is a 1-bit flag to indicate whether the message is signed with the SAC session key or not. A setting of '0' means the message is not signed with the SAC session key and no such signature is appended to the message. A setting of '1' means the message is signed with the SAC authentication session key and a 160-bit signature is appended to the message. The signature is calculated over the whole CPCM message, i.e. over the CPCM_protocol_message_type field, the control_field and all of the message_payload_bytes. The algorithm for generating CPCM message signatures is the CPCM MAC algorithm defined in TS 102 825-5 [i.8].

payload_signed_ADS: This is a 1-bit flag to indicate whether the message is signed with the AD Secret (ADS) or not. A setting of '0' means the message is not signed with the ADS and no such signature is appended to the message. A setting of '1' means the message is signed with the ADS and a 160-bit signature is appended to the message. The signature is calculated over the whole CPCM_message, i.e. over the CPCM_protocol_message_type field, the control_field and all of the message_payload_bytes. The algorithm for generating CPCM message signatures is the CPCM MAC algorithm defined in TS 102 825-5 [i.8].

A CPCM protocol message may be signed doubly with the SAC session key and with the ADS. In this case the SAC session key signature is applied to the message and appended to it as usual. ADS signature is then applied to this newly constituted message and appended to it.

payload_encrypted_SAC: This is a 1-bit flag to indicate whether the message payload is encrypted with the SAC session key or not. A setting of '0' means the message payload is not encrypted with the SAC session key. A setting of '1' means the message is encrypted with the SAC encryption session key. Encryption is applied only to the message_payload_bytes, i.e. the CPCM_protocol_message_type field and the control_field are always in the clear. The CPCM encryption algorithm applied is defined in TS 102 825-5 [i.8].

payload_encrypted_ADS: This is a 1-bit flag to indicate whether the message payload is encrypted with the AD Secret (ADS) or not. A setting of '0' means the message payload is not encrypted with the ADS. A setting of '1' means the message is encrypted with the ADS. Encryption is applied only to the message_payload_bytes, i.e. the CPCM_protocol_message_type field and the control_field are always in the clear. The CPCM encryption algorithm applied is defined in TS 102 825-5 [i.8].

A CPCM protocol message may be encrypted doubly with the SAC session key and with the ADS. In this case the SAC encryption key signature is first applied to the message. ADS encryption is then applied to the result.

NOTE: ADS encryption of CPCM protocol messages is not used in the present document.

6.2.3 Generic atomic transaction

Some CPCM protocols require the use of a transaction mechanism, so that a CPCM protocol can be backed out if an error occurs at any step in the protocol. A multi-phase 'prepare and commit' transaction mechanism is defined for this purpose.

Table 42: Generic atomic transaction message types

Message type	Function and use
Begin	Initial request to begin an atomic transaction. Sent by the requestor to the provider.
Ready	Response to a begin, indicating that the provider is able to accept the request, and is ready to carry it out. This implies that all necessary resources are available and reserved, and that any pre-requisite transactions are also ready.
Commit	Response from the requestor to the provider, instructing it to complete the transaction. Once this has been sent, the requestor has no further opportunity to abandon the transaction.
Confirm	Confirmation from the provider that the transaction has been entirely completed.
Finish	Used when necessary to ensure delivery of the confirm message.
Transaction Rollback	An Indication by either party that this transaction is being abandoned and that all resources should be returned to their original state.

The Begin, Ready, Commit, Confirm and Finish steps of a transaction are implemented with protocol-specific dedicated protocol messages defined in each case.

The Transaction Rollback is a generic CPCM protocol message used to back out of any CPCM protocol transaction.

The basic information flow for such a transaction is illustrated in figure 6.

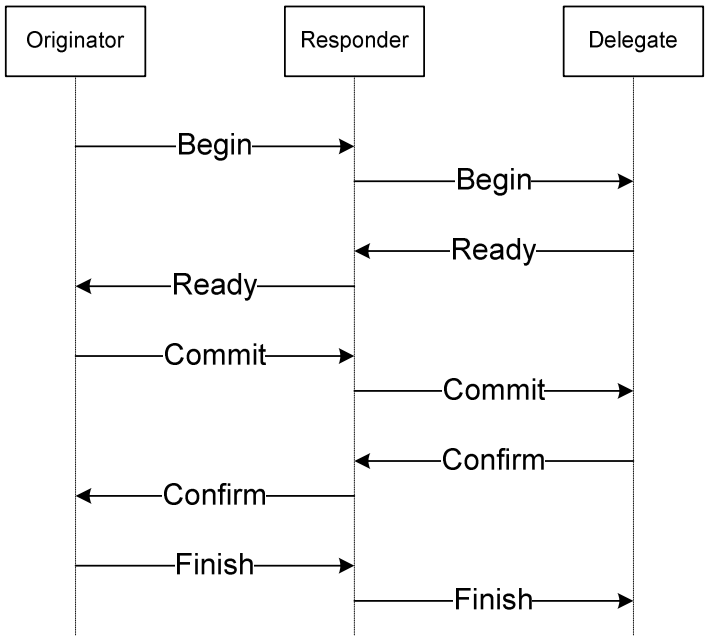

Figure 6: Basic transaction flow

The original provider to whom the transaction is directed (the "responder") is able to initiate additional transactions at one or more other providers (the "delegates"). This sequence ensures that either all entities execute the transaction to completeness, or the entire transaction is cancelled with all entities returning to their original states.

There is a slight variant of this approach, where the trigger for starting a transaction (i.e. invitation) comes from another entity, as shown in figure 7.

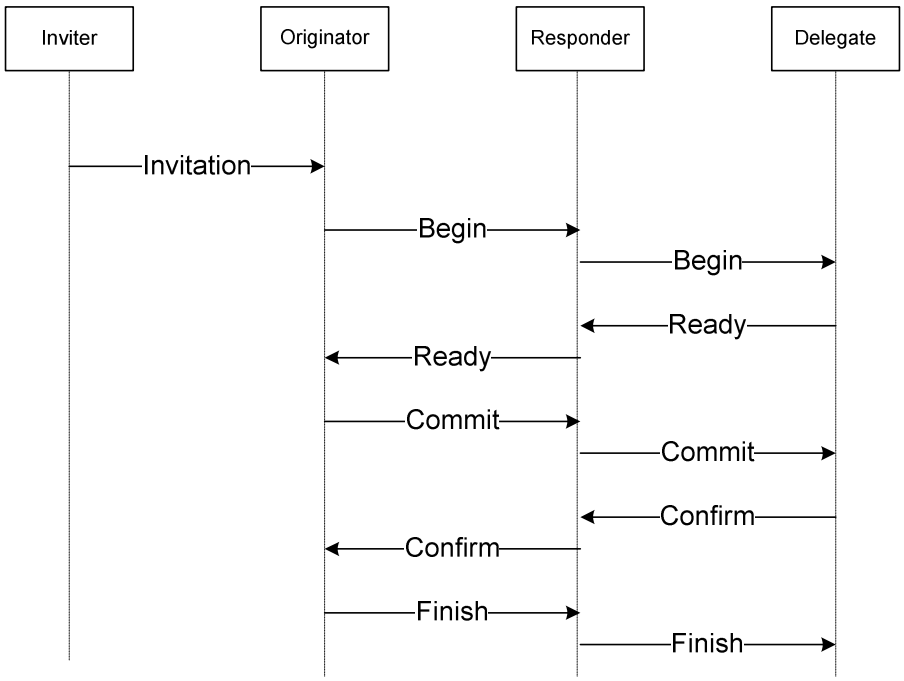

Figure 7: Invited transaction flow

In this case, an invitation to execute the transaction is sent from a provider to another entity, inviting that entity to begin a specific transaction. Assume the invitation is acted upon, the transaction proceeds in exactly the same way as in figure 6.

Figures 8 to 10 show a break down of the processing within each individual entity.

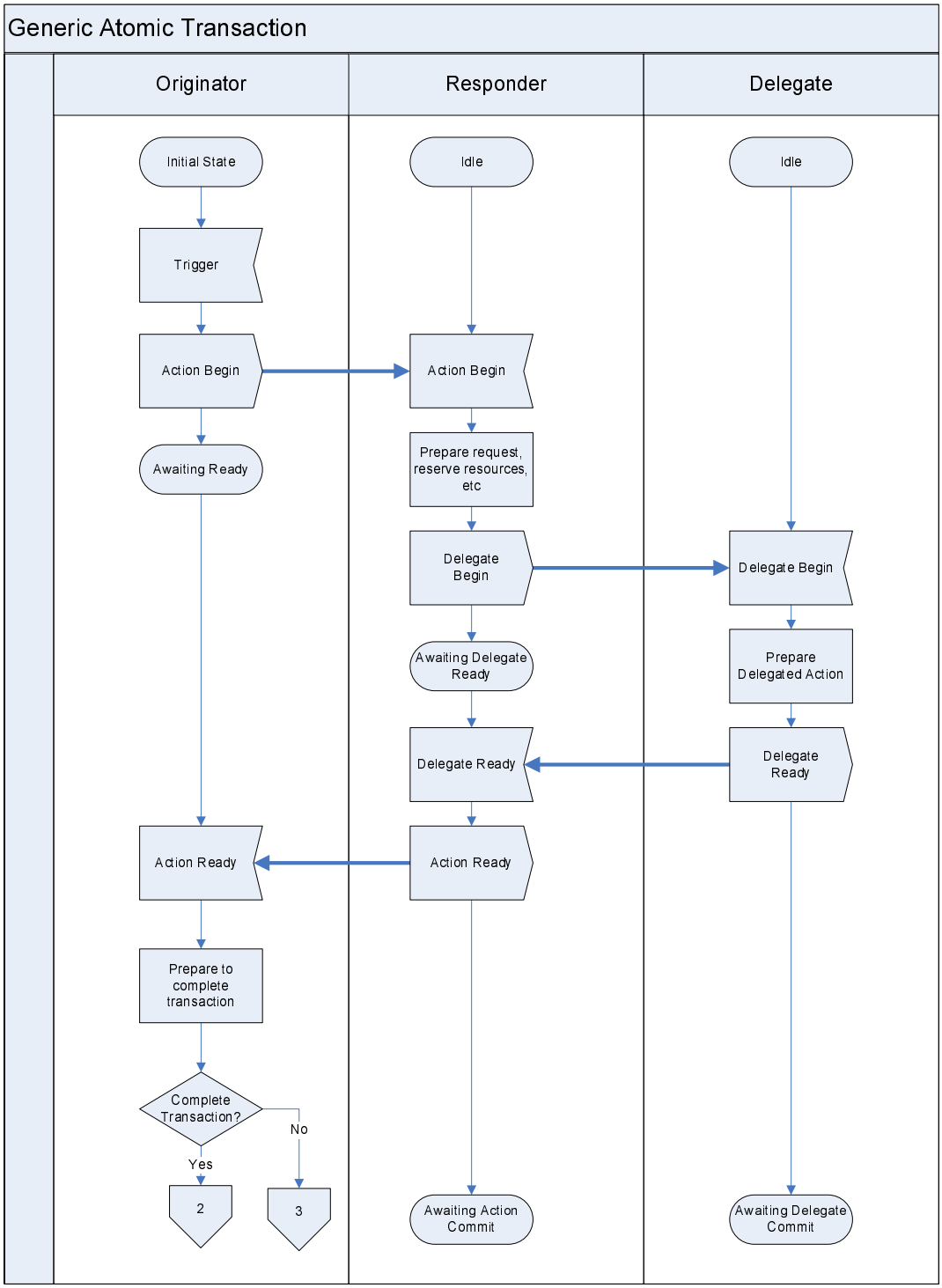

Figure 8: Atomic transaction initialization

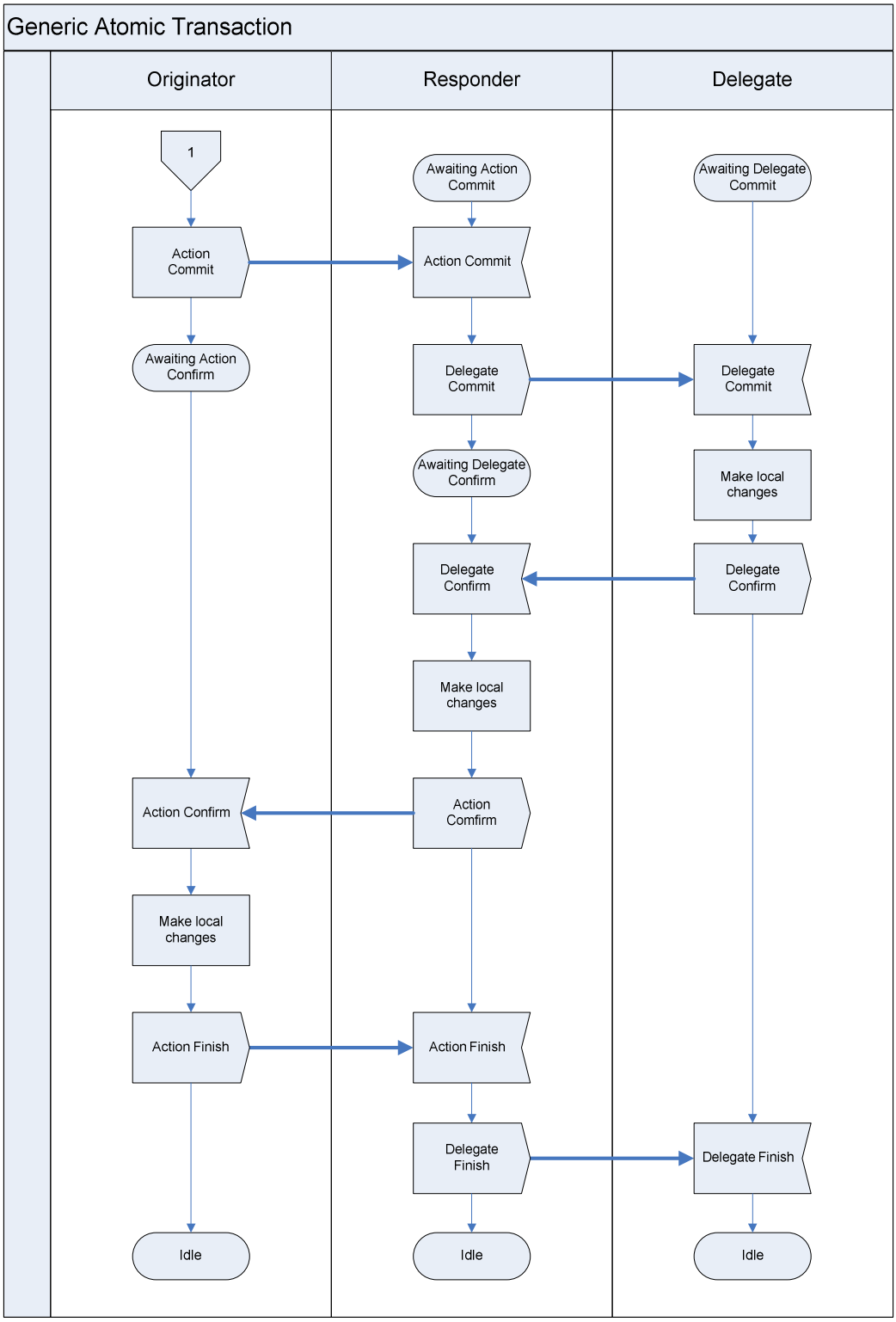

Figure 9: Atomic transaction completion

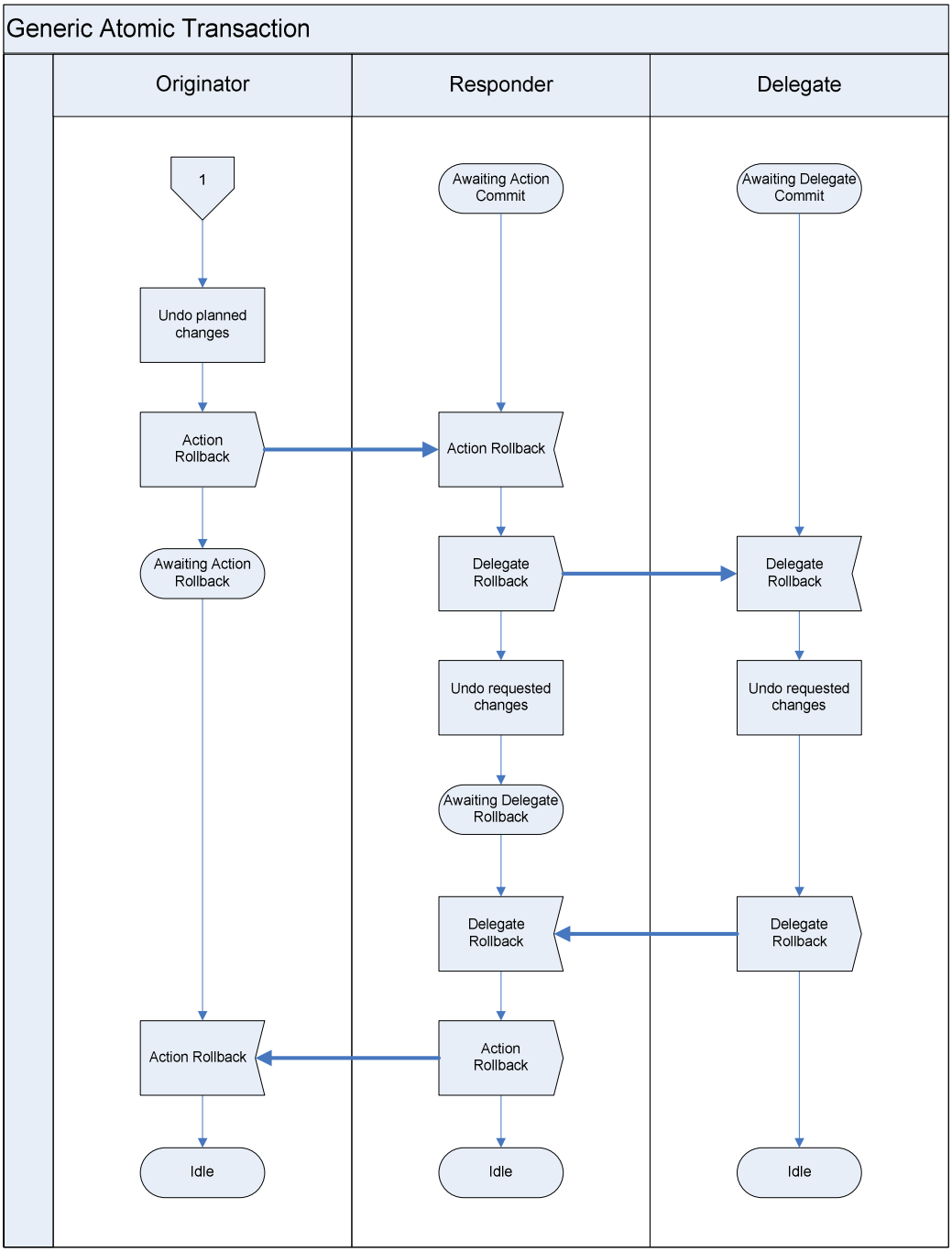

Figure 10: Atomic transaction abandonment

Figure 10 shows the originator abandoning the transaction; the transaction may also be abandoned by any entity at any point by substituting the Rollback message for the normal response. This transaction rollback message will then be propagated to all entities participating in the transaction, and the whole process will be abandoned.

It is important to note that entities must prepare as fully as possible for a transaction before returning a ready message, such that there is a very high degree of likelihood that the subsequent commit will succeed. Resources should be fully allocated and reserved and all dependent transactions also fully advanced to the ready state before reporting ready.

6.2.4 CPCM protocol timeouts

The CPCM System specification does not define any maximum time during which a CPCM Instance shall wait for a response to a message. This is left to implementation choice. Waiting duration is likely to vary regarding the nature of the sent or broadcasted message and consequently the possible actions the peer CPCM Instance shall do before being able to answer.

The different parameters that will affect the duration time include the following:

- Whether the message is sent or broadcast.
- Whether a human interaction is expected to occur.
- Whether intermediary protocols (e.g. proximity tests, quorum tests or secure data exchanges) are expected to occur.

6.2.5 CPCM protocol bridging

The CPCM System is agnostic as regards network and interface technologies, but any deployment of CPCM must apply at least one of the CPCM System adaptation layers for home network ecosystems (see TS 102 825-9 [i.3]) or application-specific physical interfaces (see TS 102 825-9 [i.3]).

If a CPCM Device deploys more than one of the home network ecosystems or application-specific physical interfaces for which a CPCM System adaptation is defined in TS 102 825-9 [i.3], and if that CPCM Device facilitates bridging between those networks and/or interfaces as regards content discovery and exchange, then the CPCM Instance in that device may also enable bridging of the CPCM System between those networks and/or interfaces. This facility might also depend on the requirements of the C&R regime applying to that CPCM Instance.

6.3 General CPCM protocol messages

6.3.1 General

The category of general CPCM protocol messages consists of:

- CPCM protocol error message; and
- CPCM protocol transaction rollback message.

Table 43 lists the protocol-specific message codes for general CPCM protocol messages.

Table 43: General CPCM protocol message codes

Message code	Message
0x00	reserved
0x01	CPCM_error_message
0x02	CPCM_transaction_rollback_message
0x03 to OxFF	reserved

6.3.2 CPCM protocol error

6.3.2.1 General

```
void CPCM_protocol_error_message (
    CPCM_protocol_message_type              message_type_request,
    error_code                              error
);
```

CPCM_protocol_error_message is a generic CPCM protocol message that is used to indicate an error condition upon a previously received CPCM protocol message.

Table 44 shows the structure of CPCM_protocol_error_message.

Table 44: CPCM protocol error message

Syntax	Bits	Identifier
CPCM_protocol_error_message(){		
CPCM_protocol_message_type = 0xFF01	16	uimsbf
control_field	8	uimsbf
message_payload_length	16	uimsbf
CPCM_protocol_message_type_request	16	CPCM_protocol_message_type
error_code	8	uimsbf
[conditional_elements()]		
[optional_elements()]		
[SAC_secret_signature]	160	CPCM_MAC
}		

Semantics for CPCM_protocol_error_message:

CPCM_protocol_message_type: This shall be set to the value corresponding to CPCM_protocol_error according to tables 40 and 43, namely 0xFF01.

control_field: See clause 6.2.2.6. If the original CPCM protocol message was SAC-signed, then the CPCM protocol error message shall also be SAC-signed. The CPCM protocol error message shall not be ADS-signed and shall not be encrypted.

message_payload_length: This shall be set to the number of message_payload_bytes of the message plus the length of the message signature, if the message is signed.

CPCM_protocol_message_type_request: This is the CPCM Message Type of the CPCM protocol message that resulted in the CPCM protocol error message.

conditional_elements: The delegating_instance_id element as defined in clause 6.6.2.5 may be contained here (see clause 6.2.2.2).

error_code: This is an 8-bit value that indicates the reason for failure of the original CPCM protocol message. For generic errors the CPCM Instance generating the CPCM protocol error message shall set this field to the appropriate value as defined in clause 6.3.2.2. For protocol-specific errors, the appropriate value for the respective protocol shall be used, as defined for each CPCM protocol in clause 6.4 onwards.

SAC_secret_signature: This field, defined in TS 102 825-5 [i.8], is present if the error message is signed.

6.3.2.2 CPCM protocol generic error codes

Error codes in response to a CPCM protocol message request are divided into two categories - generic error codes valid alike for all CPCM protocols, and protocol-specific error codes.

The values zero to 31 (0x1F) for error_code are reserved for generic error codes.

The values of 32 to 255 (0x20 to 0xFF) are reserved for protocol-specific error codes. Each CPCM protocol defines a list of protocol-specific error codes in clause 6.4 onwards.

Table 45 lists the enumerators for CPCM protocol generic error_code and their meanings.

Table 45: CPCM protocol generic message error codes

Error code	Meaning
0x00	Reserved
0x01	Unspecified error condition
0x02	Message syntax error
0x03	Protocol context error
0x04	Function not implemented
0x05	CPCM System version error
0x06	Protocol message SAC signature error
0x07	Protocol message AD signature error
0x08	Protocol timeout
0x09	SAC expired
0x0A	SAC not established
0x0B	Destination CPCM Instance not present
0x0C to 0x1F	Reserved for future use
0x20 to 0xFF	Reserved for protocol-specific error codes

6.3.3 CPCM protocol transaction rollback

```
void CPCM_protocol_transaction_rollback_message (
  CPCM_protocol                                    protocol,
  error_code                                           error
  [conditional_elements()]
  [optional_elements()]
  [SAC_secret_signature]
);
```

This message shall be sent by any Instance involved in a transaction in the case of an error or rejection. This can occur either before or after a SAC has been established between the communicating CPCM Instances, hence this affects whether or not this protocol message is signed or not, as described below.

Table 46: CPCM Transaction rollback message

Syntax	Bits	Identifier
CPCM_protocol_transaction_rollback_message(){		
CPCM_protocol_message_type = 0xFF02	16	bslbf
control_field	8	bslbf
message_payload_length	16	uimsbf
CPCM_protocol	8	bslbf
error_code	8	uimsbf
[conditional_elements()]		
[optional_elements()]		
[SAC_secret_signature]	160	CPCM_MAC
}		

NOTE: The delegating_instance_id element as defined in clause 6.2.2.4 may be present within the conditional_elements.

Semantics for the CPCM_protocol_transaction_rollback_message:

CPCM_protocol_message_type: According to tables 40 and 43 this shall be set to 0xFF02.

control_field: This message shall not be encrypted. If a SAC is already in place then the message shall be signed, hence this field shall be set to 0x08. If there is no SAC in place then the message shall not be signed, hence this field shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 2 plus the number of bytes contained in any optional elements.

CPCM_protocol: The protocol to which transaction rollback applies, as shown in table 47.

Table 47: CPCM protocol

Value	Meaning
0	Content Move
1	Join AD
2	Leave AD
3	Domain Controller Transfer
4	Domain Controller Split
5	Domain Controller Merge
6	Domain Controller Rebalance

error_code: This is an 8-bit value that indicates the reason for failure of the original CPCM transaction. For generic errors the CPCM Instance generating the CPCM protocol error message shall set this field to the appropriate value as defined in clause 6.3.2.2. For protocol-specific errors, the appropriate value for the respective protocol shall be used, as defined for the relevant protocol call, as defined in clause 6.4 onwards.

SAC_secret_signature: This field, defined in TS 102 825-5 [i.8], is present if the message is signed.

The possible CPCM protocol error codes specific to the CPCM protocol transaction rollback protocol call are listed in table 48.

Table 48: CPCM protocol transaction rollback specific error codes

Error code	Meaning
0x20	Transaction not in progress for indicated protocol
0x21 to 0xFF	Reserved for future use

6.4 CPCM security control protocols

6.4.1 General

The CPCM Security Control (SEC) protocols consist of the following functions, specified in the following clauses:

- CPCM Authenticated Key Exchange (AKE) (or SAC establishment).
- CPCM AD Secret management.

Table 49 lists the protocol-specific message codes for CPCM Security Control (SEC).

Table 49: CPCM security control protocol message codes

SEC message code	Message
0x00	Reserved
0x01	CPCM_AKE_init_message
0x02	CPCM_AKE_commit_message
0x03	CPCM_AKE_renew_message
0x04	CPCM_AKE_commit_renew_message
0x05	CPCM_AKE_confirm_message
0x06	CPCM_AKE_confirm_response
0x07	CPCM_AKE_terminate_message
0x08	Deliver_AD_secret_message
0x09	Deliver_AD_secret_reponse_message
0x0A	Erase_AD_secret_message
0x0B	Erase_AD_response_message
0x0C to OxFF	Reserved

Table 50 provides the complete list of error codes and their meanings used defined for use in the CPCM security control protocols. Not all errors are valid for each individual protocol call, so the ones that are valid for each call are listed with each protocol call definition.

Table 50: Summary of CPCM security control specific error codes

Error code	Meaning
0x20	Incorrect certificate signature
0x21	Invalid certificate
0x22	Invalid certificate chain
0x23	Certificate has been revoked
0x24	Ancestor certificate has been revoked
0x25	Certificate C&R regime mismatch
0x26	Incorrect trust check value
0x27	SAC not in place
0x28	CPCM Instance is not AD aware
0x29	CPCM Instance is not a member of any AD
0x2A	No AD Secret had been stored
0x2B to 0xFF	Reserved for future use

6.4.2 SAC establishment

6.4.2.1 General

The CPCM AKE protocol enables two CPCM Instances (A and B) to authenticate each other according to the algorithms, defined in TS 102 825-5 [i.8], and to establish a Secure Authenticated Channel, whereby nominal CPCM Instance A initiates the AKE process with nominal CPCM Instance B.

Figure 11 shows the protocol sequence for a new AKE process in order to establish a SAC between CPCM Instances A and B.

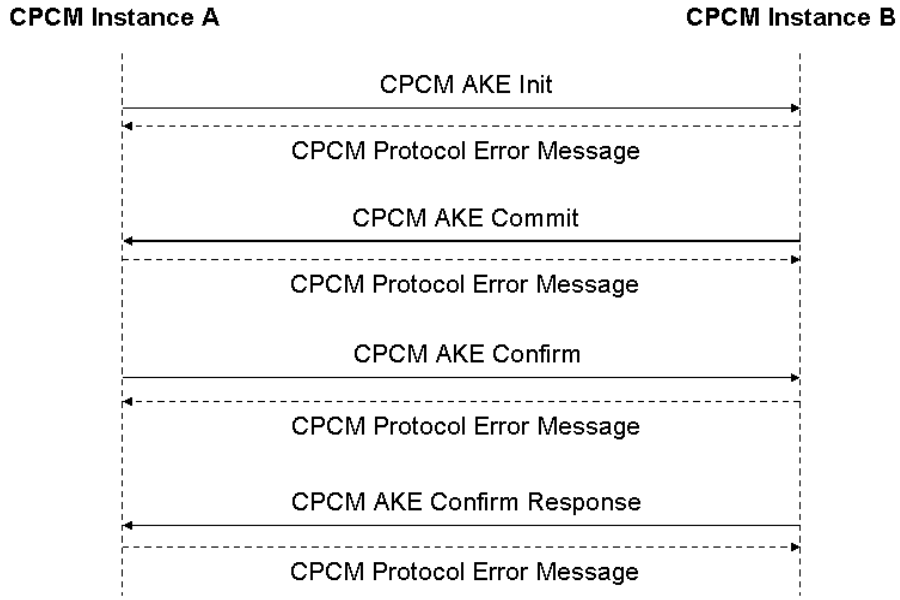

Figure 11: CPCM AKE protocol

The SAC session key shall remain valid for a maximum time (SAC key expiry time) the value of which is not set by the present document. If SAC session key expiry time occurs then one of the following shall occur:

- the SAC shall be terminated (by either partner) and the terminating CPCM Instance shall notify the SAC partner CPCM Instance by using the CPCM SAC terminate protocol; or
- the SAC shall be renewed using the CPCM AKE Renewal protocol.

SAC session timeout is depicted in figure 12, whereby the option for AKE termination is depicted.

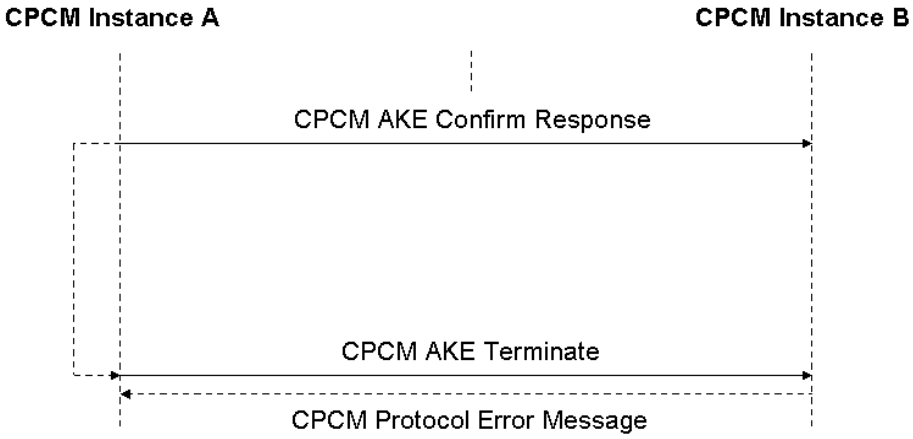

Figure 12: CPCM SAC session key expiry

The CPCM AKE Renewal protocol enables the renewal of an existing SAC between CPCM Instances A and B. This is shown in figure 13.

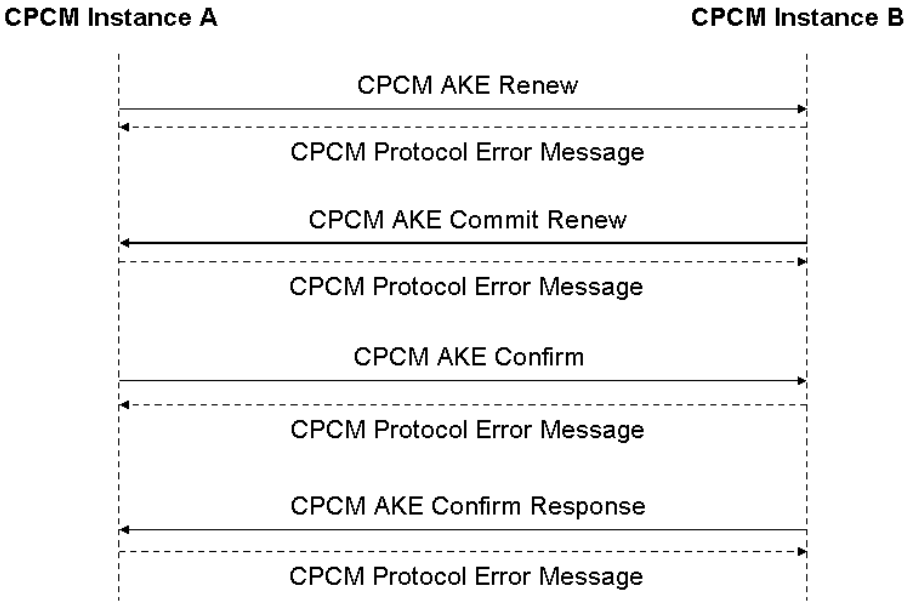

Figure 13: CPCM AKE renewal protocol

SAC renewal may be initiated in order to pre-empt the expiry of the SAC session key.

Each of the protocol steps is specified in the following clauses.

6.4.2.2 CPCM AKE initiation

```
void CPCM_AKE_init_message (
    CPCM_witness                          witness_A,
    CPCM_instance_certificate_chain       certificate_chain_A
);
```

Table 51 specifies the CPCM protocol message for CPCM AKE initiation.

Table 51: CPCM AKE initiation

Syntax	Bits	Identifier
CPCM_AKE_init_message(){		
CPCM_protocol_message_type = 0x0101	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
witness_A	1024	CPCM_witness
CPCM_instance_certificate_chain_A	n*2048+8	CPCM_instance_certificate_chain
}		

Semantics for CPCM_AKE_init_message:

CPCM_protocol_message_type: According to tables 40 and 49 this shall be set to 0x0101.

control_field: This message shall be neither signed nor encrypted; hence this shall be set to 0x00.

message_payload_length: This indicates the number of message_payload_bytes of the message.

witness_A: Diffie-Hellman public value provided by CPCM Instance A.

CPCM_instance_certificate_chain_A: The CPCM Instance Certificate chain of CPCM Instance A.

The possible CPCM protocol error codes specific to the CPCM AKE initiate protocol call are listed in table 52.

Table 52: CPCM AKE initiate specific error codes

Error code	Meaning
0x20	Incorrect certificate signature
0x21	Invalid certificate
0x22	Invalid certificate chain
0x23	Certificate has been revoked
0x24	Ancestor certificate has been revoked
0x25	Certificate C&R regime mismatch
0x26 to 0x2A	Reserved
0x2B to 0xFF	Reserved for future use

6.4.2.3 CPCM AKE commit

```
void CPCM_AKE_commit_message (
    CPCM_witness                          witness_B,
    CPCM_instance_certificate_chain       certificate_chain_B,
    CPCM_trust_check                      check_B
);
```

Table 53 specifies the CPCM protocol message for CPCM AKE commit.

Table 53: CPCM AKE commit

Syntax	Bits	Identifier
CPCM_AKE_commit_message(){		
CPCM_protocol_message_type = 0x0102	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
witness_B	1024	CPCM_witness
CPCM_instance_certificate_chain_B	n*2048+8	CPCM_instance_certificate_chain
check_B	160	CPCM_trust_check
}		

Semantics for CPCM_AKE_commit_message:

CPCM_protocol_message_type: According to tables 40 and 49 this shall be set to 0x0102.

control_field: This message shall be neither signed nor encrypted; hence this shall be set to 0x00.

message_payload_length: This indicates the number of message_payload_bytes of the message.

witness_B: Diffie-Hellman public value provided by CPCM Instance B.

CPCM_instance_certificate_chain_B: The CPCM Instance Certificate chain of CPCM Instance B.

check_B: Hash value, as defined in TS 102 825-5 [i.8], calculated by CPCM Instance B.

The possible CPCM protocol error codes specific to the CPCM AKE commit protocol call are listed in table 54.

Table 54: CPCM AKE commit specific error codes

Error code	Meaning
0x20	Incorrect certificate signature
0x21	Invalid certificate
0x22	Invalid certificate chain
0x23	Certificate has been revoked
0x24	Ancestor certificate has been revoked
0x25	Certificate C&R regime mismatch
0x26	Incorrect trust check value
0x27 to 0x2A	Reserved
0x2B to 0xFF	Reserved for future use

6.4.2.4 CPCM AKE renew

```
void CPCM_AKE_renew_message (
    CPCM_witness                          witness_A,
    CPCM_instance_certificate_id          instance_id_A
);
```

Table 55 specifies the CPCM protocol message for CPCM AKE renew.

Table 55: CPCM AKE renew

Syntax	Bits	Identifier
CPCM_AKE_renew_message(){		
CPCM_protocol_message_type = 0x0103	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
witness_A	1 024	CPCM_witness
CPCM_instance_certificate_id_A	64	CPCM_instance_certificate_id
}		

Semantics for CPCM_AKE_renew_message:

CPCM_protocol_message_type: According to tables 40 and 49 this shall be set to 0x0103.

control_field: This message shall be neither signed nor encrypted; hence this shall be set to 0x00.

message_payload_length: This indicates the number of message_payload_bytes of the message; hence this shall be set to 136.

witness_A: Diffie-Hellman public value provided by CPCM Instance A.

CPCM_instance_certificate_id_A: The CPCM Instance Identifier of CPCM Instance A.

The possible CPCM protocol error codes specific to the CPCM AKE renew protocol call are listed in table 56.

Table 56: CPCM AKE renew specific error codes

Error code	Meaning
0x20 to 0x26	Reserved
0x27	SAC not in place
0x28 to 0x2A	Reserved
0x2B to 0xFF	Reserved for future use

6.4.2.5 CPCM AKE commit renew

```
void CPCM_AKE_commit_renew_message (
    CPCM_witness                              witness_B,
    CPCM_instance_certificate_id              instance_id_B,
    CPCM_trust_check                          check_B
);
```

Table 57 specifies the CPCM protocol message for CPCM AKE commit renew.

Table 57: CPCM AKE commit renew

Syntax	Bits	Identifier
CPCM_AKE_renew_message(){		
CPCM_protocol_message_type = 0x0104	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
witness_B	1 024	CPCM_witness
CPCM_instance_certificate_id_B	64	CPCM_instance_certificate_id
check_B	160	CPCM_trust_check
}		

Semantics for CPCM_AKE_commit_renew_message:

CPCM_protocol_message_type: According to tables 40 and 49 this shall be set to 0x0104.

control_field: This message shall be neither signed nor encrypted; hence this shall be set to 0x00.

message_payload_length: This indicates the number of message_payload_bytes of the message; hence this shall be set to 156.

witness_B: Diffie-Hellman public value provided by CPCM Instance B.

CPCM_instance_certificate_id_B: The CPCM Instance Identifier of CPCM Instance B.

check_B: Hash value, as defined in TS 102 825-5 [i.8], calculated by CPCM Instance B.

The possible CPCM protocol error codes specific to the CPCM AKE commit renew protocol call are listed in Table 58.

Table 58: CPCM AKE commit renew specific error codes

Error code	Meaning
0x20-0x25	reserved
0x26	incorrect trust check value
0x27-0x2A	reserved
0x2B-0xFF	reserved for future use

6.4.2.6 CPCM AKE confirm

```
void CPCM_AKE_confirm (
    CPCM_trust_check                  check_A
);
```

Table 59 specifies the CPCM protocol message for CPCM AKE confirm.

Table 59: CPCM AKE Confirm

Syntax	Bits	Identifier
CPCM_AKE_confirm_message(){		
CPCM_protocol_message_type = 0x0105	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
check_A	160	CPCM_trust_check
}		

Semantics for CPCM_AKE_confirm_message:

CPCM_protocol_message_type: According to tables 40 and 49 this shall be set to 0x0105.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: This indicates the number of message_payload_bytes of the message; hence this shall be set to 20.

check_A: Hash value, as defined in TS 102 825-5 [i.8], calculated by CPCM Instance A.

The possible CPCM protocol error codes specific to the CPCM AKE confirm protocol call are listed in Table 60.

Table 60: CPCM AKE confirm specific error codes

Error code	Meaning
0x20 to 0x25	Reserved
0x26	Incorrect trust check value
0x27 to 0x2A	Reserved
0x2B to 0xFF	Reserved for future use

6.4.2.7 CPCM AKE confirm response

```
void CPCM_AKE_confirm_response_message ( void );
```

Table 61 specifies the CPCM protocol message for CPCM AKE confirm response.

Table 61: CPCM AKE confirm response

Syntax	Bits	Identifier
CPCM_AKE_confirm_response_message(){		
CPCM_protocol_message_type = 0x0106	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_AKE_confirm_response_message:

CPCM_protocol_message_type: According to tables 40 and 49 this shall be set to 0x0106.

control_field: This message shall be signed using the SAC key but not encrypted, hence this shall be set to 0x08.

message_payload_length: This indicates the number of message_payload_bytes of the message; hence this shall be set to 20.

SAC_secret_signature: This is the message signature as specified in TS 102 825-5 [i.8].

There are no protocol-specific error codes defined for this protocol call.

6.4.2.8 CPCM AKE terminate

```
void CPCM_AKE_terminate_message ( void );
```

Table 62 specifies the CPCM protocol message for CPCM AKE terminate.

Table 62: CPCM AKE Terminate

Syntax	Bits	Identifier
CPCM_AKE_termninate_message(){		
CPCM_protocol_message_type = 0x0107	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_AKE_terminate_message:

CPCM_protocol_message_type: According to tables 40 and 49 this shall be set to 0x0107.

control_field: This message shall be signed using the SAC key but not encrypted, hence this shall be set to 0x08.

message_payload_length: This indicates the number of message_payload_bytes of the message; hence this shall be set to 0.

SAC_secret_signature: This is the message signature as specified in TS 102 825-5 [i.8].

The possible CPCM protocol error codes specific to the CPCM AKE terminate protocol call are listed in table 63.

Table 63: CPCM AKE terminate specific error codes

Error code	Meaning
0x20 to 0x26	Reserved
0x27	SAC not in place
0x28 to 0x2A	Reserved
0x2B to 0xFF	Reserved for future use

6.4.3 CPCM AD Secret management

6.4.3.1 General

The CPCM AD secret management protocol is used in the following circumstances:

- to provide the AD secret to a CPCM Instance that has successfully joined an AD, as shown in figure 14; and
- to ensure that the AD secret has been erased by a CPCM Instance that has left the AD, as shown in figure 15.

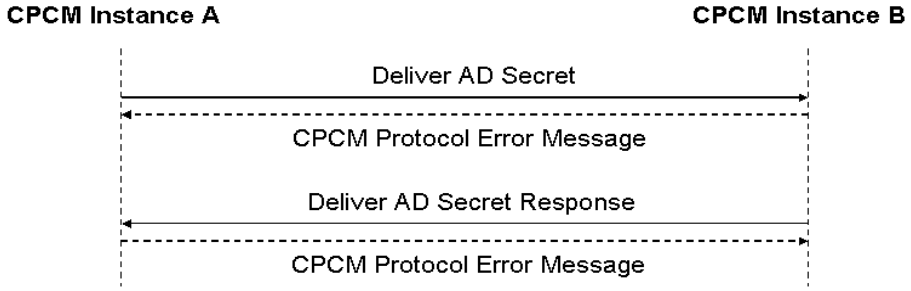

Figure 14: AD Secret delivery protocol

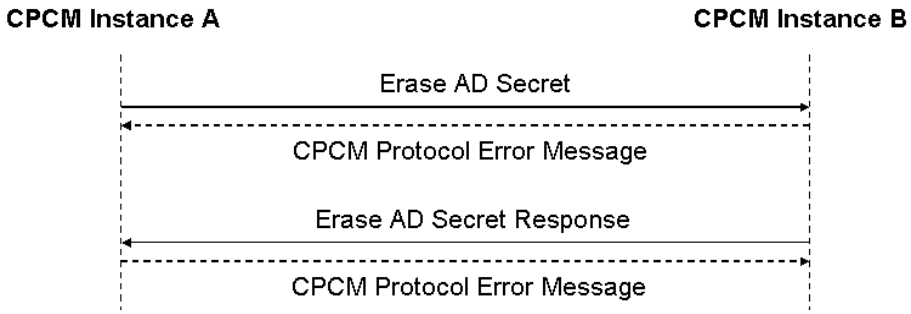

Figure 15: AD secret erasure protocol

6.4.3.2 Deliver AD Secret

```
void deliver_AD_secret_message {
    AD_secret                   secret
};
```

The *deliver_AD_secret_message* is used to provide the AD Secret to a CPCM Instance that has joined an AD.

Table 64: Deliver AD secret message

Syntax	Bits	Identifier
deliver_AD_secret_message(){		
CPCM_protocol_message_type = 0x0108	16	bslbf
control_field = 0x0A	8	bslbf
message_payload_length	16	uimsbf
AD_secret	128	bslbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for deliver_AD_secret_message:

CPCM_protocol_message_type: According to tables 40 and 49 this shall be set to 0x0108.

control_field: This message shall be both signed and encrypted using SAC session keys, hence this shall be set to 0x0A.

message_payload_length: This indicates the number of message_payload_bytes of the message; hence this shall be set to 36.

AD_secret: This is the 128-bit AD Secret.

SAC_secret_signature: This is the message signature as specified in TS 102 825-5 [i.8].

The possible CPCM protocol error codes specific to the deliver AD Secret protocol call are listed in table 65.

Table 65: Deliver AD Secret specific error codes

Error code	Meaning
0x20 to 0x27	Reserved
0x28	CPCM Instance is not ADM capable
0x29	CPCM Instance is already a member of an AD
0x2A	Reserved
0x2B to 0xFF	Reserved for future use

6.4.3.3 Deliver AD Secret response

```
void deliver_AD_secret_response_message ( void );
```

The *deliver_AD_secret_response_message* is used by the newly joined CPCM Instance to confirm the delivery of the AD Secret.

Table 66: Deliver AD Secret response message

Syntax	Bits	Identifier
deliver_AD_secret_reponse_message(){		
CPCM_protocol_message_type = 0x109	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for deliver_AD_secret_reponse_message:

CPCM_protocol_message_type: According to tables 40 and 49 this shall be set to 0x0109.

control_field: This message shall be signed using SAC session key, hence this shall be set to 0x08.

message_payload_length: This indicates the number of message_payload_bytes of the message; hence this shall be set to 20.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8].

There are no protocol-specific error codes defined for this protocol call.

6.4.3.4 Erase AD Secret

```
void erase_AD_secret_message ( void );
```

The *erase_AD_secret_message* is used to command a CPCM Instance that has left an AD to erase the AD Secret for that AD.

Table 67: Erase AD secret message

Syntax	Bits	Identifier
erase_AD_secret_message(){		
CPCM_protocol_message_type = 0x010A	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for erase_AD_secret_message:

CPCM_protocol_message_type: According to tables 40 and 49 this shall be set to 0x010A.

control_field: This message shall be signed using SAC session key, hence this shall be set to 0x08.

message_payload_length: This indicates the number of message_payload_bytes of the message; hence this shall be set to 20.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8].

The possible CPCM protocol error codes specific to the erase AD Secret protocol call are listed in table 68.

Table 68: Erase AD Secret specific error codes

Error code	Meaning
0x20 to 0x27	Reserved
0x28	CPCM Instance is not ADM capable
0x29	CPCM Instance is not a member of any AD
0x2A	No AD Secret had been stored
0x2B to 0xFF	Reserved for future use

6.4.3.5 Erase AD Secret response

```
void erase_AD_secret_response_message ( void );
```

The *erase_AD_secret_response_message* is used to confirm the erasure of the AD Secret.

Table 69: Erase AD secret response message

Syntax	Bits	Identifier
erase_AD_secret_response_message(){		
CPCM_protocol_message_type = 0x010B	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for erase_AD_secret_response_message:

CPCM_protocol_message_type: According to tables 40 and 49 this shall be set to 0x010B.

control_field: This message shall be signed using SAC session key, hence this shall be set to 0x08.

message_payload_length: This indicates the number of message_payload_bytes of the message; hence this shall be set to 20.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8].

There are no protocol-specific error codes defined for this protocol call.

6.5 CPCM System and Content Management Protocols

6.5.1 General

The CPCM System and Content Management (SYS) protocols consist of the following functions, specified in the following sub-sections:

- 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.

Table 70 lists the protocol-specific message codes for CPCM System and Content Management (SYS).

Table 70: CPCM System protocol message codes

SYS message code	Message
0x00	Reserved
0x01	CPCM_instance_status_enquiry_message
0x02	CPCM_instance_status_enquiry_response_message
0x03	CPCM_get_CL_message
0x04	CPCM_get_CL_response_message
0x05	CPCM_get_new_CL_message
0x06	CPCM_put_CL_message
0x07	CPCM_put_CL_response_message
0x08	CPCM_get_permission_message
0x09	CPCM_get_permission_response_message
0x0A	CPCM_get_content_item_status_request_message
0x0B	CPCM_get_content_item_status_responses
0x0C	CPCM_get_absolute_time_message
0x0D	CPCM_get_absolute_time_response
0x0E	CPCM_AD_membership_challenge_message
0x0F	CPCM_AD_membership_challenge_response_message
0x10	CPCM_get_geo_location_message
0x11	CPCM_get_geo_location_response_message
0x12	CPCM_enquire_geo_location_formats_message
0x13	CPCM_enquire_geo_location_formats_response_message
0x14	CPCM_affirm_geo_location_message
0x15	CPCM_affirm_geo_location_response_message
0x16	CPCM_get_RL_message
0x17	CPCM_notify_RL_message
0x18	CPCM_CL_move_begin_message
0x19	CPCM_CL_move_ready_message
0x1A	CPCM_CL_move_commit_message
0x1B	CPCM_CL_move_confirm_message
0x1C	CPCM_CL_move_finish_message
0x1D to Ox1F	Reserved
0x20	CPCM_discovery_request_message
0x21	CPCM_discovery_response_message
0x22 to 0x2F	Reserved
0x30	CPCM_rtt_request_message
0x31	CPCM_rtt_response_message
0x32	CPCM_srtt_request_message
0x33	CPCM_srtt_response_message
0x34	CPCM_srtt_validation_message
0x35	CPCM_srtt_validation_response_message
0x36	CPCM_PTA_proximity_test_request_message
0x37	CPCM_PTA_proximity_test_response_message
0x38	CPCM_AAA_proximity_test_assignment_request_message
0x39	CPCM_AAA_proximity_test_assignment_response_message
0x3A to 0xFF	Reserved

Table 71 provides the complete list of error codes and their meanings used defined for use in the CPCM security control protocols. Not all errors are valid for each individual protocol call, so the ones that are valid for each call are listed with each protocol call definition.

Table 71: Summary of CPCM System and Content management specific error codes

Error code	Meaning
0x20	Unknown Content Licence identifier
0x21	Content handling operation refused
0x22	Invalid original Content Licence
0x23	Grant of new USI refused
0x24	Inappropriate content handling operation
0x25	Invalid permission type
0x26	Content handling operation mismatch
0x27	Content Licence syntax error
0x28	Content Licence signature error
0x29 to 0x2F	Reserved
0x30	RTT too high: proximity test aborted
0x31	Request challenge mismatch: proximity test failure
0x32	Response challenge mismatch: proximity test failure
0x33	SRTT request TTL too low: proximity test failure
0x34	SRTT response TTL too low: proximity test failure
0x35 to 0x3F	Reserved
0x40 to 0x5F	Reserved for adaptation layer specific proximity test error codes
0x60	CPCM Instance is not secure time aware
0x61	Absolute time currently not available
0x62 to 0x6F	Reserved
0x70	CPCM Instance is not geographic aware
0x71	Geographic location format not supported
0x72	Geographic location information currently not available
0x73 to 0x7F	Reserved
0x80	CPCM Instance is not AD aware
0x81	CPCM Instance is not a member of any AD
0x82	CPCM Instance is not a member of the indicated AD
0x83	The indicated AD has been revoked
0x84	Request challenge mismatch: AD membership not confirmed
0x85	Response challenge mismatch: AD membership not confirmed
0x86 to 0x8F	Reserved
0x90 to 0xFF	Reserved for future use

6.5.2 CPCM instance status

6.5.2.1 General

This protocol is used by a CPCM Instance to enquire the current status of another CPCM Instance. The CPCM status enquiry response contains static information from the CPCM Instance Certificate and dynamic information like ADM status variables (if applicable).

The CPCM status enquiry protocol is shown in figure 16.

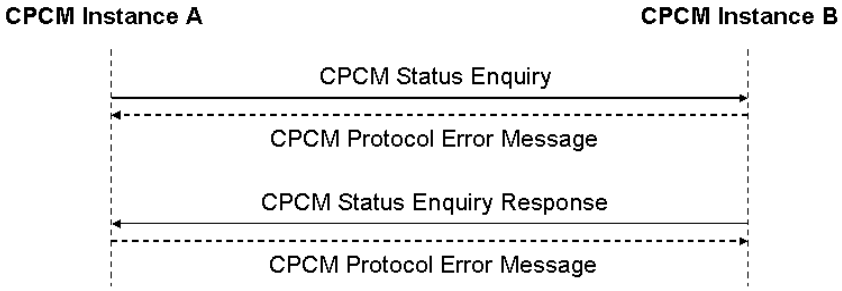

Figure 16: CPCM status enquiry Protocol

6.5.2.2 CPCM Instance status enquiry

```
void CPCM_instance_status_enquiry_message ( void );
```

Table 72 specifies the CPCM protocol message for CPCM Instance status enquiry.

Table 72: CPCM Instance status enquiry

Syntax	Bits	Identifier
CPCM_instance_status_enquiry_message(){		
CPCM_protocol_message_type = 0x4001	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
}		

Semantics for CPCM_instance_status_enquiry_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4001.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 0.

There are no protocol-specific error codes defined for this protocol call.

6.5.2.3 CPCM Instance status enquiry response

```
void CPCM_instance_status_enquiry_response_message (
    CPCM_status_information     instance_CPCM_status,
    ADM_status_information      instance_ADM_status
);
```

Table 73 specifies the CPCM protocol message for CPCM Instance status enquiry response.

Table 73: CPCM Instance status enquiry response

Syntax	Bits	Identifier
CPCM_instance_status_enquiry_response_message(){		
CPCM_protocol_message_type = 0x4002	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
instance_CPCM_status()	104	CPCM_status_information
instance_ADM_status()	8	ADM_status_information
}		

Semantics for CPCM_instance_status_enquiry_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4002.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message.

instance_CPCM_status: Structure containing status information about the responding CPCM Instance (see clause 5.4.6).

instance_ADM_status: Structure containing ADM-relevant status information about the responding CPCM Instance (see clause 6.6.2.2), if applicable.

There are no protocol-specific error codes defined for this protocol call.

6.5.3 CPCM Content Licence exchange

6.5.3.1 General

CPCM Content Licence (CL) exchange can be performed in two different ways, namely in CL pull or push modes.

CPCM_get_CL (pull mode) is used when CPCM Instance wishing to receive a CPCM Content item requests the associated CPCM Content Licence for that Content item from the CPCM Instance advertising the availability of that Content item.

If successful then the requested CPCM Content Licence is returned in a *CPCM_get_CL_response _message*.

The CPCM Content Licence pull protocol is shown in figure 17.

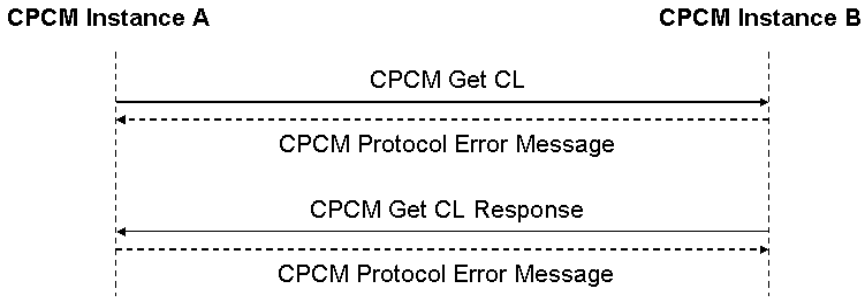

Figure 17: CPCM Content Licence pull protocol

A variant of the Content Licence pull protocol is used in order to obtain a new version of an existing Content Licence for a particular CPCM Content item, 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. Its application is described further in clause 8.5. If the addressed CPCM Instance is able to fulfil the request then it shall provide the new version of the CL using the CPCM Content Licence Push protocol, described in figure 18.

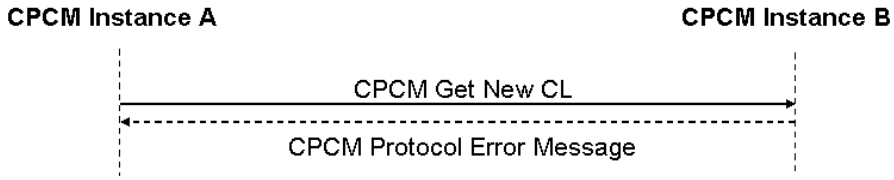

Figure 18: CPCM get new Content Licence protocol

CPCM_put_CL_message (push mode) is used for the following purposes:

- To provide a CPCM Content Licence to a Sink Device in advance of CPCM Content transfer; and
- To force a change of the content encryption key.

The *CPCM_put_CL_response_message* confirms the receipt of the passed CL.

The CPCM Content Licence push protocol is shown in figure 19.

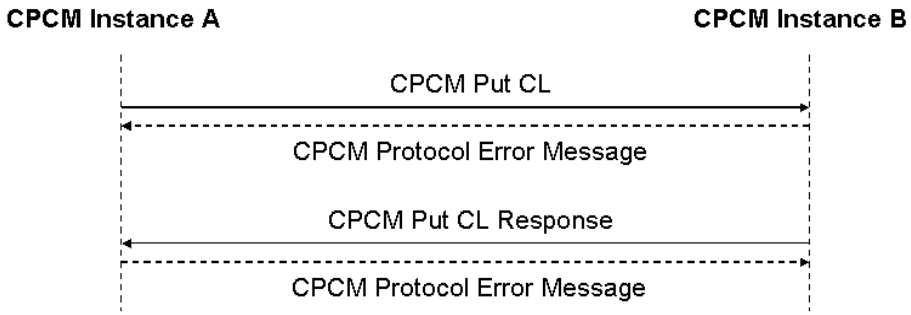

Figure 19: CPCM Content Licence push protocol

Each of the CPCM Content Licence exchange protocol steps is specified in the following clauses.

6.5.3.2 CPCM get Content Licence

```
void CPCM_get_CL_message (
    CPCM_content_licence_id                 cl_id,
    CPCM_content_handling_operation         operation
    [CPCM_instance_certificate_id_sink      sink_instance]
);
```

Table 74 specifies the CPCM protocol message for CPCM get Content Licence.

Table 74: CPCM get Content Licence

Syntax	Bits	Identifier
CPCM_get_CL_message(){		
CPCM_protocol_message_type = 0x4003	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
content_licence_id	128	uimsbf
CPCM_content_handling_operation	8	bslbf
[CPCM_instance_certificate_id_sink]	64	bslbf
}		

Semantics for CPCM_get_content_licence_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4003.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 17.

content_licence_id: CPCM Content Licence Identifier of the requested CPCM Content Licence.

CPCM_content_handling_operation: This field indicates which CPCM Content Handling operation(s) the calling CPCM Instance will perform with the Content item. Its syntax is defined in clause 5.4.7.

sink_instance: This field is conditional. Its presence depends on the content handling operation being performed. Its usage is specified in clause 7.5.

The possible CPCM protocol error codes specific to this protocol call are listed in table 75.

Table 75: CPCM get Content Licence specific error codes

Error code	Meaning
0x20	Unknown Content Licence identifier
0x21	Content handling operation refused

6.5.3.3 CPCM get Content Licence response

```
void CPCM_get_CL_response_message (
CPCM_content_handling_operation            operation,
CPCM_content_licence                       cl
);
```

This message is used to provide the requested CL, if the corresponding CL request could be fulfilled. Table 76 specifies the CPCM protocol message for CPCM get Content Licence response.

Table 76: CPCM Get Content Licence response

Syntax	Bits	Identifier
CPCM_get_CL_response_message(){		
CPCM_protocol_message_type = 0x4004	16	bslbf
control_field	8	bslbf
message_payload_length	16	uimsbf
CPCM_content_handling_operation	8	bslbf
content_licence		CPCM_content_licence
[SAC_secret_signature]	160	CPCM_MAC
}		

Semantics for CPCM_get_content_licence_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4004.

control_field: This message shall be signed and encrypted when the CPCM Content Licence requires SAC protection, hence this shall be set to 0x0A; otherwise it shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 1 plus the length of the CL passed.

CPCM_content_handling_operation: This field indicates which of the requested CPCM Content Handling operation(s) are authorized for the calling CPCM Instance to perform with the Content item. If all requested operations are authorized then this field shall have the same setting as in the *CPCM_get_CL_message* call. If not all operations requested are authorized then the authorized subset of the requested operations shall be indicated. Its syntax is defined in clause 5.4.7.

content_licence: The requested CPCM Content Licence.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8], present only if control_field is 0x0A.

The possible CPCM protocol error codes specific to this protocol call are listed in table 77.

Table 77: CPCM get Content Licence response specific error codes

Error code	Meaning
0x26	Content handling operation mismatch
0x27	Content Licence syntax error
0x28	Content Licence signature error

6.5.3.4 CPCM get new Content Licence

```
void CPCM_get_new_CL (
    CPCM_content_licence                   cl,
    CPCM_content_handling_operation        operation,
    [CPCM_USI                              USI]
);
```

Table 78 specifies the CPCM protocol message for CPCM get new Content Licence.

Table 78: CPCM get new Content Licence

Syntax	Bits	Identifier
CPCM_get_new_CL_message(){		
CPCM_protocol_message_type = 0x4005	16	bslbf
control_field = 0x0A	8	bslbf
message_payload_length	16	uimsbf
CPCM_content_handling_operation	8	bslbf
content_licence		CPCM_content_licence
[requested_usage_state_information()]		CPCM_usage_state_information
[SAC_secret_signature]	160	CPCM_MAC
}		

Semantics for CPCM_get_new_content_licence_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4005.

control_field: This message shall be signed and encrypted when the CPCM Content Licence requires SAC protection, hence this shall be set to 0x0A; otherwise it shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to n.

CPCM_content_handling_operation: This field indicates which CPCM Content Handling operation(s) the calling CPCM Instance intends to perform with the Content item. Its syntax is defined in clause 5.4.7.

content_licence: The CPCM Content Licence that needs to be re-acquired.

requested_usage_state_information: This field is optional. If present then it indicates the USI settings that the requesting CPCM Instance would like to obtain for the corresponding CPCM Content item.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8], present only if control_field is 0x0A.

The possible CPCM protocol error codes specific to the CPCM get new Content Licence protocol call are listed in table 79.

Table 79: CPCM get new Content Licence specific error codes

Error code	Meaning
0x21	Content handling operation refused
0x22	Invalid original Content Licence
0x23	Grant of new USI refused

6.5.3.5 CPCM put Content Licence

```
void CPCM_put_CL_message (
    CPCM_content_licence                    cl,
    CPCM_content_handling_operation         operation
    [CPCM_instance_certificate_id_source    source_instance]
);
```

Table 80 specifies the CPCM protocol message for CPCM put Content Licence.

Table 80: CPCM put Content Licence

Syntax	Bits	Identifier
CPCM_put_CL_message(){		
CPCM_protocol_message_type = 0x4006	16	bslbf
control_field	8	bslbf
message_payload_length	16	uimsbf
CPCM_content_handling_operation	8	bslbf
content_licence		CPCM_content_licence
[CPCM_instance_certificate_id_source]	64	bslbf
[SAC_secret_signature]	160	CPCM_MAC
}		

Semantics for CPCM_put_CL_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4006.

control_field: This message shall be signed and encrypted when the CPCM Content Licence requires SAC protection, hence this shall be set to 0x0A; otherwise it shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message.

CPCM_content_handling_operation: This field indicates which CPCM Content Handling operation(s) the Sink CPCM Instance is authorized to perform with the Content item. Its syntax is defined in clause 5.4.7.

content_licence: The CPCM Content Licence being passed.

source_instance: This field is conditional. Its presence depends on the content handling operation being performed. Its usage is specified in clause 7.3.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8], present only if control_field is 0x0A.

The possible CPCM protocol error codes specific to this protocol call are listed in table 81.

Table 81 : CPCM put Content Licence specific error codes

Error code	Meaning
0x24	Inappropriate content handling operation
0x27	Content Licence syntax error
0x28	Content Licence signature error

6.5.3.6 CPCM put Content Licence response

```
void CPCM_put_CL_response_message (void);
```

This message shall be used to confirm the reception of the CL that was passed in the corresponding *CPCM_put_CL_message*. Table 82 specifies the CPCM protocol message for CPCM put Content Licence response.

Table 82: CPCM put Content Licence response

Syntax	Bits	Identifier
CPCM_put_CL_response_message(){		
CPCM_protocol_message_type = 0x4007	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
}		

Semantics for CPCM_put_CL_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4007.

control_field: This message shall be neither signed nor encrypted hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 0.

There are no protocol-specific error codes defined for this protocol call.

6.5.4 CPCM Content operation permission

6.5.4.1 General

The CPCM Content operation permission protocol is used when a CPCM Instance already has the Content Licence for an operation but needs to verify with a CPCM Instance other than the one that supplied the Content Licence that the desired operation is allowed. The circumstances when this is necessary are documented in clause 7 and TR 102 825-11 [i.7].

The CPCM Content operation permission protocol is depicted in figure 20. It consists of two calls - the request for permission for a particular CPCM Content operation, and the corresponding response, either granting or denying permission for the operation. Alternatively, permission for the content operation may be granted permanently by CPCM Instance B issuing a new Content License, using the *CPCM_put_CL_message*, as its response.

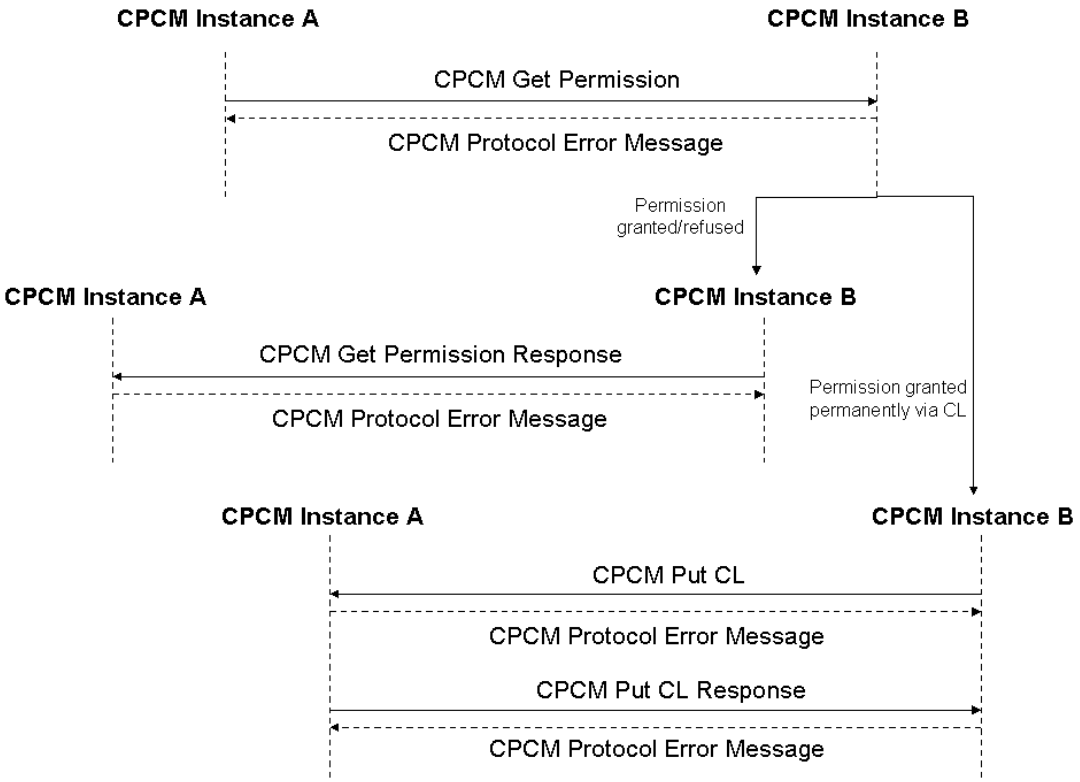

Figure 20: CPCM get permission protocol

6.5.4.2 CPCM get permission

```
void CPCM_get_permission_message (
    CPCM_content_licence_id                 cl_id,
    CPCM_content_handling_operation         operation,
    CPCM_permission_type                        permission
);
```

Table 83 specifies the CPCM protocol message for CPCM get permission.

Table 83: CPCM get permission

Syntax	Bits	Identifier
CPCM_get_permission_message(){		
CPCM_protocol_message_type = 0x4008	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
content_licence_id	128	CPCM_content_licence_identifier
CPCM_content_handling_operation	8	bslbf
permission_type	8	bslbf
}		

Semantics for CPCM_get_permission_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4008.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 18.

content_licence_id: CPCM Content Licence Identifier of the CPCM Content item for which the permission is sought.

CPCM_content_handling_operation: This field indicates which CPCM Content Handling operation(s) the calling CPCM Instance requests to perform with the Content item. Its syntax is defined in clause 5.4.7.

CPCM_permission_type: This field indicates whether the permission is requested to assess a particular case. Permission type encoding is shown in table 84.

Table 84: CPCM permission types

Message type category identifier	Message type category
0x00	unspecified type
0x01	SVCA request
0x02	remote_access_post_record request
0x03 to 0xFF	reserved

The possible CPCM protocol error codes specific to the CPCM get permission protocol call are listed in Table 85.

Table 85 : CPCM get permission specific error codes

Error code	Meaning
0x20	Unknown Content Licence identifier
0x21	Content handling operation refused
0x24	Inappropriate content handling operation
0x25	Invalid permission type

6.5.4.3 CPCM get permission response

```
void CPCM_get_permission_response_message (
    CPCM_content_licence_id                 cl_id,
    CPCM_content_handling_operation         operation
);
```

Table 86 specifies the CPCM protocol message for CPCM get permission response.

Table 86: CPCM get permission response

Syntax	Bits	Identifier
CPCM_get_permission_response_message (){		
CPCM_protocol_message_type = 0x4009	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
content_licence_id	128	CPCM_content_licence_identifier
CPCM_content_handling_operation	8	bslbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_get_permission_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4009.

control_field: This message shall be signed using SAC session key, hence this shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 37.

content_licence_id: CPCM Content Licence Identifier of the CPCM Content item to which the permissions apply.

CPCM_content_handling_operation: This field indicates which CPCM Content Handling operation(s) the calling CPCM Instance is authorized to perform. Its syntax is defined in clause 5.4.7.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8].

There are no protocol-specific error codes defined for this protocol call.

6.5.5 CPCM Content item status protocol

6.5.5.1 General

The CPCM Content item status protocol is used when a CPCM Instance needs to know whether another CPCM Instance is currently handling a CPCM Content item. The circumstances when this is necessary are documented in clause 7 and TR 102 825-11 [i.7].

The CPCM Content operation permission protocol is depicted in figure 21. It consists of two calls - the request for status for a particular CPCM Content item, and the corresponding response.

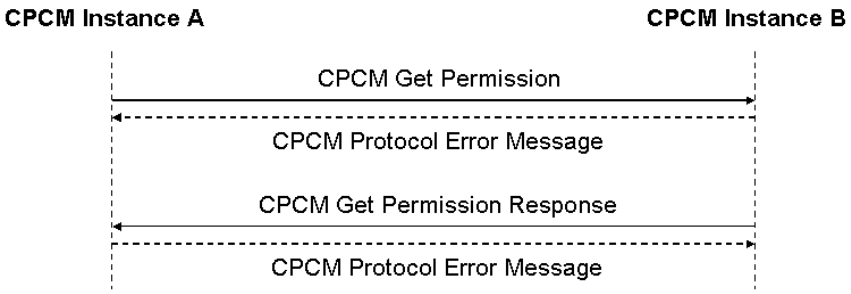

Figure 21: CPCM Content item status protocol

6.5.5.2 CPCM get Content item status request

```
void CPCM_get_content_item_status_request (
    CPCM_content_licence_id                         cl_id,
);
```

Table 87 specifies the CPCM protocol message for CPCM Get Content item status request.

Table 87: CPCM Get Content item status message

Syntax	Bits	Identifier
CPCM_get_content_item_status_request_message(){		
CPCM_protocol_message_type = 0x400A	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
content_licence_id	128	uimsbf
}		

Semantics for CPCM_get_content_item_status_request_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x400A.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 16.

content_licence_id: CPCM Content Licence Identifier of the CPCM Content item for which the status is sought.

The possible CPCM protocol error codes specific to the CPCM get Content item status protocol call are listed in table 88.

Table 88 : CPCM get Content item status specific error codes

Error code	Meaning
0x20	Unknown Content Licence identifier

6.5.5.3 CPCM get Content item status response

```
void CPCM_get_content_item_status_response_message (
    CPCM_content_licence_id                    cl_id,
    CPCM_content_handling_operation            operation
);
```

Table 89 specifies the CPCM protocol message for CPCM get Content item status response.

Table 89: CPCM get Content item status response

Syntax	Bits	Identifier
CPCM_get_content_item_status_response_message(){		
CPCM_protocol_message_type = 0x400B	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
content_licence_id	128	uimsbf
CPCM_content_handling_operation	8	bslbf
}		

Semantics for CPCM_get_content_item_status_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x400B.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 17.

content_licence_id: CPCM Content Licence Identifier of the CPCM Content item whose status is requested.

CPCM_content_handling_operation: This field indicates which CPCM Content Handling operation(s) the called CPCM Instance currently performing on the Content item. Its syntax is defined in clause 5.4.7.

There are no protocol-specific error codes defined for this protocol call.

6.5.6 CPCM Device Proximity Checks

6.5.6.1 General

The Proximity Test Method and Proximity Tools provide the means to distinguish Local from Remote for Content Usage, AD Size, Scope and Extent management and other security uses in CPCM. Local is defined as "within the immediate vicinity, approximating to the physical extent of a domicile or vehicle". Remote is "not Local".

The Proximity Test is defined as a means to determine whether two CPCM Devices, or a CPCM Device and a non-CPCM device storing CPCM Content, are Local with respect to each other at the time the test is performed. A CPCM device needs to determine whether another device is Local or not in two cases:

- Content is marked as MLAD or VLAD. Based on these settings, content propagation is only authorized within the LAD. A proximity test needs to be performed before content is transmitted to another AD instance.
- Content is marked as MLocal or VLocal. Based on these settings, content propagation is authorized within the Local Environment. A proximity test needs to be performed before content is transmitted to another device.

The Proximity Test can also be used for other CPCM administrative matters, e.g. as part of the AD Management and AD Size, Scope, Extent (ADSE) methods and tools.

Proximity Control Communications can be realized at various layers in networking protocol stacks and are often dependent upon the network interfaces deployed within the particular home network ecosystem. Hence some specifications for Proximity Control Communications are contained only in TS 102 825-9 [i.3].

6.5.6.2 Proximity method description

The Proximity Method described herein comprises a logical combination of the Proximity Tools.

The method to assess proximity comprises passing at least one of the following tests prior to any action or operation upon Content for which a Proximity restriction applies. If one of the following tests returns a logical true value indicating "Localness" of a Source to a Sink, the two will be considered "Local" to one another, regardless of a previous failure of the same or another test.

The method, therefore, comprises a logical OR of the following tests:

1) SRTT (secured round-trip time) unless otherwise defined in TS 102 825-9 [i.3], if both Source and Sink are CPCM Instances.
2) RTT (round-trip time) if only one of the Source and the Sink is a CPCM Instance.
3) PTDC (proximity through direct connection).
4) GTTP (GPS or terrestrial triangulation for proximity).
5) PAAAA (proximity assignment through authorized authenticated agent). This method may not be used in combination with another method.
6) PTA (proximity through association) where Source and Sink assess their proximity to a single "Intermediary Device" using one of the combinations of the above tests as shown in TS 102 825-9 [i.3].

NOTE: Some of the component tools of these methods have persistence and therefore do not need to be repeated for all applications of the method.

Table 90: Possible PTA combinations

Source / Sink	SRTT (Sink to Int)	RTT (Sink to Int)	PTDC (Sink to Int)	GTTP (Sink to Int)
SRTT (Source to Intermediary)	N	N	Y	Y
RTT (Source to Intermediary)	N	N	Y	Y
PTDC (Source to Intermediary)	Y	Y	Y	Y
GTTP (Source to Intermediary)	Y	Y	Y	Y

There may be situations in which a CPCM Device has multiple means of implementing some of the Proximity Tools, e.g. if two or more different physical networks can be used to interconnect two devices like wireless Ethernet and wired Ethernet. In these cases, for the general case, RTT or SRTT tools may be used on either network, or may be repeated for each network with the results being logically ORed together for the purposes of the Proximity Method.

The compliance regime may require that CPCM Devices that are capable of implementing multiple Proximity Tools give precedence to more reliable tools, with a logical "false" of a test overriding a logical "true" of another less reliable test. Any such variations from the strict logical OR of the method described above shall be specifically called out by the compliance regime for this particular Device, or class of Devices as may be defined by the compliance regime.

Table 91 give a list of identifiers for the different proximity tests methods.

Table 91: CPCM Proximity methods

Proximity method identifier	Proximity method
0x00	reserved
0x30	RTT
0x31	SRTT
0x32	GTTP
0x33	PTA
0x34	PAAAA
0x35	PTDC
0x36 to 0x5F	reserved
0x60 to 0x7F	reserved for adaptation layers

Additional Methods may be defined by the compliance regime.

6.5.6.3 Proximity Tools

6.5.6.3.1 Round Trip Time protocol (RTT)

6.5.6.3.1.1 General

The Round Trip Time test protocol consists of measuring the time required to receive a response to a ping request on the network under test. The protocol sequence is shown in figure 22.

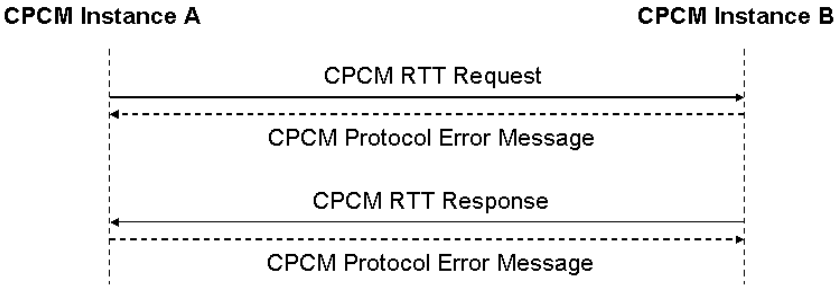

Figure 22: Round trip time protocol

This test may only be used when a CPCM Instance wishes to test whether a non-CPCM instance is Local or Remote.

The RTT tool shall use the following secure data fields:

- Network_type which will be protected against unauthorized modification. Multiple network_type fields may be defined.
- RTT_maximum_time which will be independently set for each network_type and which will be protected against unauthorized modification.
- RTT_persistence which will be protected against unauthorized modification.

Two devices (a Source and a Sink) will be considered "Local" with respect to one another if:

a) They are interconnected using network types described by network_type.

b) They have performed an RTT test with a result of RTT_maximum_time for that network_type or less within the past RTT_persistence.

Tests can be repeated multiple times and a negative test will not undo a positive test. The passage of time from the last positive test can change a "Local" indication to a "Remote" indication with respect to this RTT test.

An RTT_maximum_time for each different network_type shall be defined by the CPCM C&R regime. The CPCM Instance must use the RTT_maximum_time associated with the network_type of the particular physical network interface, e.g. PHY and LINK types, over which the RTT test is being conducted.

RTT_persistence shall be defined by C&R regime.

6.5.6.3.1.2 CPCM RTT request

```
void CPCM_rtt_request_message ( void );
```

Table 92 specifies the CPCM protocol message for CPCM RTT request.

Table 92: CPCM RTT request

Syntax	Bits	Identifier
CPCM_rtt_request_message(){		
CPCM_protocol_message_type = 0x4030	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
}		

Semantics for CPCM_rtt_request_message:

CPCM_protocol_message_type: According to tables 40 and 70, this shall be set to 0x4030.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 0.

There are no protocol-specific error codes defined for this protocol call.

6.5.6.3.1.3 CPCM RTT response

```
void CPCM_rtt_response_message ( void );
```

Table 93 specifies the CPCM protocol message for CPCM RTT response.

Table 93: CPCM RTT response

Syntax	Bits	Identifier
CPCM_rtt_response_message(){		
CPCM_protocol_message_type = 0x4031	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
}		

Semantics for CPCM_rtt_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4031.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 0.

The possible CPCM protocol error codes specific to the CPCM RTT response protocol call are listed in table 94.

Table 94 : CPCM RTT response specific error codes

Error code	Meaning
0x30	RTT too high: proximity test aborted

6.5.6.3.2 Secured Round Trip Time protocol (SRTT)

6.5.6.3.2.1 General

The SRTT Test may only be used when a CPCM Instance wishes to test whether another CPCM Instance is Local or Remote. The SRTT test starts in the same manner as its non-secure counterpart, RTT, with two exceptions: i) pings and pings responses each include a random number and ii) the total allowed round trip time is specifically tailored with a priori knowledge of the physical networks that are used by the Tester and the Testee. The "Tester" is the CPCM Instance that initiates the SRTT. The "Testee" is the CPCM Instance that responds to the Tester. Figure 23 depicts the SRTT protocol.

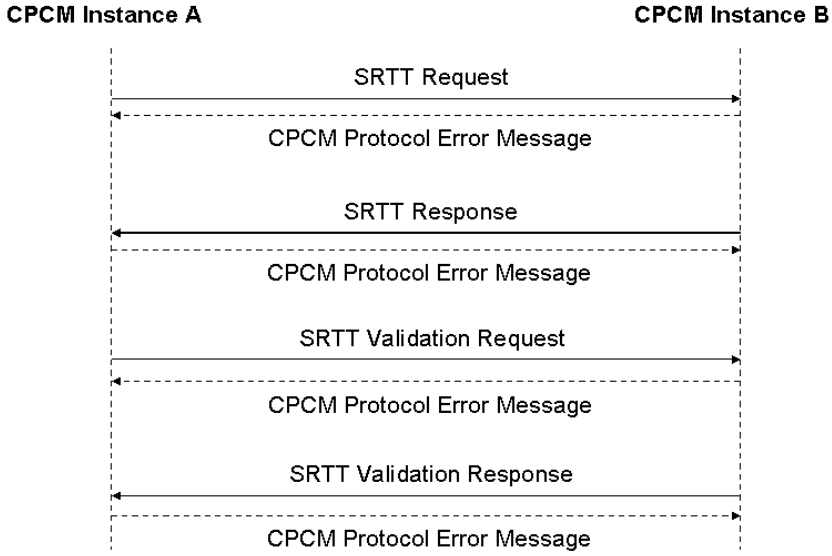

Figure 23: SRTT protocol

The SRTT tool shall use the following secure data fields:

- Network_Type_Tester which will be protected against unauthorized modification. Multiple Network_Type_Tester fields may be defined.
- SRTT_Maximum_Time which will be independently set for each Network_Type_Tester and which will be protected against unauthorized modification.
- SRTT_Persistence which will be protected against unauthorized modification.
- SRTT_Adjustment_Value_Testee which will be protected against unauthorized modification.

For SRTT, the total allowed time is the sum of the SRTT_Maximum_Time and the SRTT_Adjustment_Value_Testee, if any. This allows the a priori knowledge of the Testee's physical network interface to influence the time that the Tester allocates to the round trip time test. Some network may include a heterogeneous mixture of networking technologies and the Tester may be limited to securely knowing only its particular physical network interface type.

The SRTT_Adjustment_Value is a 24-bit 2's complement number with the least significant bit having the value 2 nanoseconds allowing the number to represent approximately ±16 milliseconds.

For example:

- 0x000000 = 0.
- 0xFFFFFF = -2 nsec.

- 0x000001 = +2 nsec.
- 0x7FFFFF = ~ +16 msec.
- 0x800000 = ~ -16 msec.

The random number included in the pings and responses shall be new for each ping or ping response and may be prepared in advance for low-resource devices.

As in the RTT tests, pings are sent until a response arrives within a given period of time (SRTT_Maximum_Time + SRTT_Adjustment_Value_Testee) and are only valid for time SRTT_Persistence. Each response from the Testee includes the SRTT_Adjustment_Value_Testee in the SRTT ready message. When device CPCM Instance Tester receives a response from CPCM Instance Testee within (SRTT Maximum_Time + SRTT_Adjustment_Value_Testee) it sends back a message to Testee asking for ping authentication. Testee answers back with a MAC computed on both random values that were included in the ping and ping response and the SRTT_Adjustment_Value. The key used for the MAC is by default the SAC session key. The key used for MAC shall be the AD Secret when explicitly stated in the relevant part of the present document.

6.5.6.3.2.2 SRTT request

```
void CPCM_srtt_request_message (
challenge                       random_A
);
```

Table 95 specifies the CPCM protocol message for CPCM SRTT request.

Table 95: CPCM SRTT request

Syntax	Bits	Identifier
CPCM_srtt_request_message(){		
CPCM_protocol_message_type = 0x4032	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
challenge	128	bslbf
}		

Semantics for CPCM_srtt_request_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4032.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 16.

challenge: A random number generated by the calling CPCM Instance to later validate the RTT.

There are no protocol-specific error codes defined for this protocol call.

6.5.6.3.2.3 SRTT response

```
void CPCM_srtt_response_message (
challenge                       random_B,
SRTT_adjustment_value       value
);
```

Table 96 specifies the CPCM protocol message for CPCM SRTT response.

Table 96: CPCM SRTT response

Syntax	Bits	Identifier
CPCM_srtt_response_message(){		
CPCM_protocol_message_type = 0x4033	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
challenge	128	bslbf
SRTT_adjustment_value	24	bslbf
}		

Semantics for CPCM_srtt_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4033.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 16.

challenge: A random number generated by the sending CPCM Instance to later validate the RTT.

SRTT_adjustment_value: An adaptation value used to cope with the different network technologies (see clause 6.5.6.3.2.1).

The possible CPCM protocol error codes specific to the CPCM SRTT response protocol call are listed in table 97.

Table 97: CPCM SRTT response specific error codes

Error code	Meaning
0x30	RTT too high: proximity test aborted

6.5.6.3.2.4 SRTT validation request

```
void CPCM_srtt_validation_request_message ( void );
```

Table 98 specifies the CPCM protocol message for CPCM RTT request.

Table 98: CPCM SRTT validation request

Syntax	Bits	Identifier
CPCM_srtt_validation_request_message(){		
CPCM_protocol_message_type = 0x4034	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
}		

Semantics for CPCM_srtt_validation_request_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4034.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 0.

There are no protocol-specific error codes defined for this protocol call.

6.5.6.3.2.5 SRTT validation response

```
void CPCM_srtt_validation_response_message (
challenge                       random_A,
challenge                       random_B,
SRTT_adjustment_value           value
);
```

Table 99 specifies the CPCM protocol message for CPCM SRTT validation response.

Table 99: CPCM SRTT Validation response

Syntax	Bits	Identifier
CPCM_srtt_validation_response_message(){		
CPCM_protocol_message_type = 0x4035	16	bslbf
control_field = 0x08 or 0x04	8	bslbf
message_payload_length	16	uimsbf
random_A	128	bslbf
random_B	128	bslbf
SRTT_adjustment_value	24	bslbf
SAC_secret_signature || AD_secret_signature	160	bslbf
}		

Semantics for CPCM_srtt_validation_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4035.

control_field: This message shall be either signed by SAC session key or the AD Secret, hence it shall be either 0x08 or 0x04.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 55.

random_A: The same random number as sent the SRTT request.

random_B: The same random number as sent the SRTT response.

SRTT_adjustment_value: An adaptation value used to cope with the different network technologies (see clause 6.5.6.3.2.1.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8], present only if control_field is 0x08.

ADS_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8], present only if control_field is 0x04.

The possible CPCM protocol error codes specific to the CPCM SRTT validation response protocol call are listed in table 100.

Table 100: CPCM SRTT validation response specific error codes

Error code	Meaning
0x31	Request challenge mismatch: proximity test failure
0x32	Response challenge mismatch: proximity test failure

6.5.6.3.3 Re-using GPS or Terrestrial Triangulation for Proximity (GTTP)

The GTTP tool shall use the data field, GTTP_Distance, which must be protected against unauthorized modification.

The following method specifies a method to determine whether Device A is Local to Device B.

This test is only usable if both devices have GTTP.

Device A uses GPS or a Terrestrial Triangulation technique to determine its location to the accuracy of the particular technology, e.g. 10 m, 20 m, 50 m.

Device B uses GPS or a Terrestrial Triangulation technique to determine its location to the accuracy of the particular technology, e.g. 10 m, 20 m, 50 m.

Devices A and B exchange their locations with one another using the SAC or another secure channel.

The format of the location information shall be as specified in clause 5.4.10. Messages used to exchange this information or *CPCM_get_geographic_location_information_message* and or *CPCM_get_geographic_location_information_response_message*.

Either or both Devices subtract the location coordinates to determine the distance.

Distances of less than GTTP_Distance will be treated as Local.

The Devices share and compare the results. A non-contradicted test of Local will indicate to both CPCM Devices that they are Local to one another.

C&R regime will establish requirements for situations in which these devices lose the triangulation signal, e.g. limited persistence of the last location determined.

For this tool, the geographic area must have been determined directly by each respective CPCM Instance through self-geographic awareness and not by interacting with another Local CPCM Instance.

GTTP_Distance will be defined by C&R regime.

6.5.6.3.4 Proximity Through Association (PTA)

6.5.6.3.4.1 General

For certain proximity tests, if A is Local to B and B is Local to C, then A is Local to C, regardless of whether or not it can be confirmed through one of the above tests. A and B have to be CPCM Instances while C is not necessarily a CPCM Instance. The protocols to be applied when C is not a CPCM Instances are described in TS 102 825-9 [i.3].

Certain proximity tests may be subject to circumvention by application of PTA once or by recursive application of PTA. In such cases PTA will not apply.

To use that test, a CPCM Instance sends in a message the CIC identifier of a CPCM Instance which is Local and the method it used to assess that. The receiving CPCM Instance assesses whether it is Local to the third CPCM Instance using a proximity test that can be used in conjunction with PTA. It sends back the result to the requesting Instance. The protocol sequence is shown in figure 24.

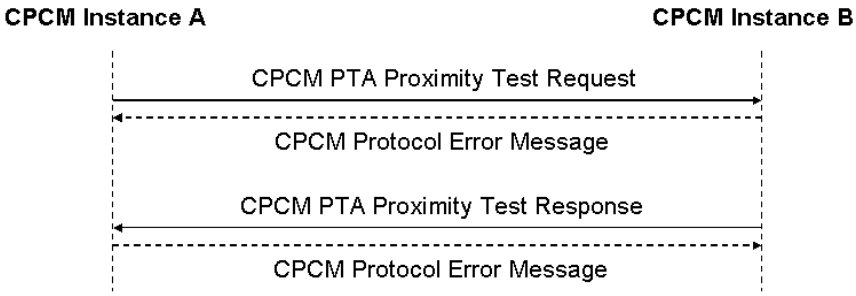

Figure 24: PTA protocol

6.5.6.3.4.2 PTA Proximity test request

```
void CPCM_PTA_proximity_test_request_message(
    CPCM_instance_certificate_id            id
    proximity_method                        method
);
```

Table 101 specifies the CPCM protocol message for CPCM PTA proximity test request.

Table 101: CPCM PTA proximity test request

Syntax	Bits	Identifier
CPCM_PTA_proximity_test_request_message(){		
CPCM_protocol_message_type = 0x4036	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
CPCM_instance_certificate_id	64	bslbf
proximity_method	8	bslbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_PTA_proximity_test_request_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4036.

control_field: This message shall be signed but not encrypted, hence this shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message, hence it shall be set to 29.

CPCM_instance_certificate_id: CIC identifier of the CPCM Instance with which the calling CPCM Instance is Local.

proximity_method: Proximity test that was used by the calling CPCM Instance as shown in table 91.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8].

The possible CPCM protocol error codes specific to the CPCM PTA proximity test request protocol call are listed in table 102.

Table 102: CPCM PTA proximity test request specific error codes

Error code	Meaning
0x33	Unknown proximity test method

6.5.6.3.4.3 PTA Proximity Test response

```
void CPCM_PTA_proximity_test_response_message(
PTA_proximity__affirmation                  affirmation
);
```

Table 103 specifies the CPCM protocol message for CPCM Affirm Geographic Location response.

Table 103: CPCM PTA Proximity test response

Syntax	Bits	Identifier
CPCM_PTA_proximity_test_response_message(){		
CPCM_protocol_message_type = 0x4037	16	bslbf
control_field	8	bslbf
message_payload_length	16	uimsbf
PTA_proximity_affirmation	8	bslbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_PTA_proximity_test_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4037.

control_field: This message shall be signed but not encrypted, hence this shall be set to 0x08

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 21.

PTA_proximity_affirmation: Code to indicate the affirmation of the proximity test. A value of 0x00 indicates that the CPCM Instance is not currently Local with the tested one. A value of 0x01 indicates that the CPCM Instance is currently Local with the tested one.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8].

There are no protocol-specific error codes defined for this protocol call.

6.5.6.3.5 Proximity Assignment by Authorized Authenticated Agent (PAAAA)

6.5.6.3.5.1 General

Localness between two devices may be assigned by an Authorized Authenticated Agent as a failsafe.

The PAAAA tool shall use a data field named PAAAA_Persistence, which must be protected against unauthorized modification. This field requires the assignment to be refreshed after a period of time equal to PAAAA_Persistence has elapsed.

This test requires both devices to be CPCM compliant.

To use that tool, each CPCM Instance sends a request to the AAA that makes the assessment and replies to each of the requesting CPCM Instances. The protocol sequence is depicted in figure 25.

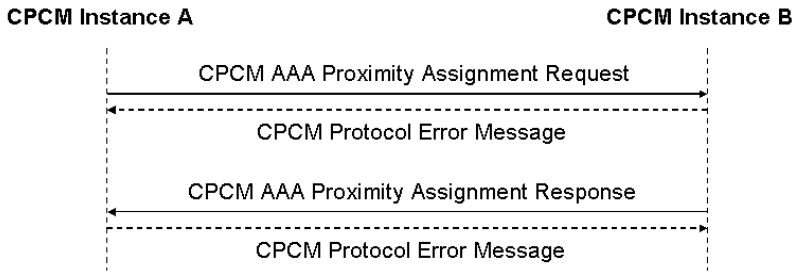

Figure 25: AAA proximity assignment protocol

6.5.6.3.5.2 AAA Proximity assignment request

```
void CPCM_AAA_proximity_assignment_request_message(
    CPCM_instance_certificate_id           id
);
```

Table 104 specifies the CPCM protocol message for CPCM AAA proximity assignment request.

Table 104: CPCM AAA proximity assignment request

Syntax	Bits	Identifier
CPCM_AAA_proximity_assignment_request_message(){		
CPCM_protocol_message_type = 0x4038	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
CPCM_instance_certificate_id	64	Bsblf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_AAA_proximity_assignment_request_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4038.

control_field: This message shall be signed but not encrypted, hence this shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message, hence it shall be set to 28.

CPCM_instance_certificate_id: CIC identifier of the CPCM Instance with which proximity shall be tested.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8].

There are no protocol-specific error codes defined for this protocol call.

6.5.6.3.5.3 AAA Proximity assignment response

```
void CPCM_AAA_proximity_assignment_response_message(
    CPCM_instance_certificate_identifier           id
    CPCM_AAA_proximity__affirmation                affirmation
);
```

Table 105 specifies the CPCM protocol message for CPCM Affirm Geographic Location response.

Table 105: CPCM AAA Proximity assignment response

Syntax	Bits	Identifier
CPCM_AAA_proximity_assignment_response_message(){		
CPCM_protocol_message_type = 0x4039	16	bslbf
control_field	8	bslbf
message_payload_length	16	uimsbf
CPCM_instance_certificate_id	64	bslbf
AAA_proximity_affirmation	8	bslbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_AAA_proximity_assignment_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4039.

control_field: This message shall be signed but not encrypted, hence this shall be set to 0x08

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 29.

CPCM_instance_certificate_id: CIC identifier of the CPCM Instance with which proximity was tested.

AAA_proximity_affirmation: Code to indicate the affirmation of the proximity test. A value of 0x00 indicates that the CPCM Instance is not currently Local to the tested one. A value of 0x01 indicates that the CPCM Instance is currently Local to the tested one.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8].

There are no protocol-specific error codes defined for this protocol call.

6.5.6.3.6 Proximity through direct connection (PTDC)

If A can robustly communicate with B through a direct physical connection with very low electrical latency, then A and B are Local. In the case of a smartcard, for example, this method would be valid upon the existence of a method ensuring that a smart card has been effectively plugged into device A.

This test requires both devices to be CPCM compliant.

PTDC methods will be further defined in TS 102 825-9 [i.3].

6.5.6.4 Mandatory Proximity tools

The following Proximity tools are mandatory for all CPCM Instances:

- SRTT between two CPCM Instances.
- RTT between a CPCM Instance and a non-compliant device.

6.5.7 CPCM secure time

6.5.7.1 General

The CPCM secure time protocol is used by a CPCM Instance that does not implement secure time itself to obtain the current secure absolute time from a CPCM Instance that does implement secure time, for example when required in order to be able to handle a CPCM Content item that has time-based USI applied to it.

The CPCM secure time protocol is shown in figure 26. It consists of two protocol steps - to request the current absolute, and the corresponding response that passes the absolute time.

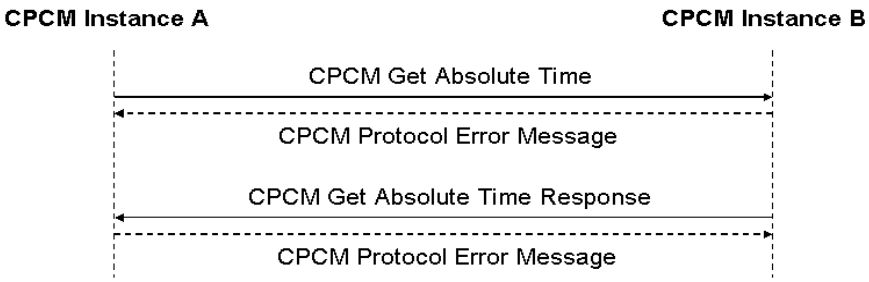

Figure 26: CPCM secure time protocol

6.5.7.2 CPCM get absolute time

```
void CPCM_get_absolute_time_message ( void );
```

Table 106 specifies the CPCM protocol message for CPCM get absolute time.

Table 106: CPCM get absolute time

Syntax	Bits	Identifier
CPCM_get_absolute_time_message(){		
CPCM_protocol_message_type = 0X400C	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
}		

Semantics for CPCM_get_absolute_time_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x400C.

control_field: This message shall neither be signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 0.

The possible CPCM protocol error codes specific to the CPCM get absolute time protocol call are listed in table 107.

Table 107 : CPCM get absolute time specific error codes

Error code	Meaning
0x60	CPCM Instance is not secure time aware
0x61	Absolute time currently not available

6.5.7.3 CPCM get absolute time response

```
void CPCM_get_absolute_time_response_message (
    CPCM_date_time          date_time
);
```

Table 108 specifies the CPCM protocol message for CPCM get absolute time response.

Table 108: CPCM Get Absolute Time response

Syntax	Bits	Identifier
CPCM_get_absolute_time_response_message(){		
CPCM_protocol_message_type = 0x400D	16	bslbf
Control_field	8	bslbf
Message_payload_length	16	uimsbf
date_time	40	CPCM_date_time
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_get_absolute_time_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x400D.

control_field: This message shall be signed but not encrypted; hence this shall be set to 0x08.

message_payload_length: the number of message_payload_bytes of the message, hence this shall be set to 25.

date_time: The absolute current date and time as maintained by the responding CPCM Instance.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8].

There are no protocol-specific error codes defined for this protocol call.

6.5.8 Geographic information

6.5.8.1 General

The CPCM geographic information protocol is used by a CPCM Instance that is not geographically aware itself to verify with another CPCM Instance within the Local Environment that is geographically aware whether both CPCM Instances are currently located within the region signalled by the passed geographic location information.

There are two alternative methods for passing geographic location information. The first method assumes that the device that is geographically aware also verifies, or affirms, that the CPCM Instance is currently located within the provided geographic area. The second method is applied when the device that is geographically aware just provides the current geographic location, in one of the defined formats, and does not perform any kind of evaluation of the location.

There is also a query protocol defined to enable the discovery of the supported geographic location formats for both of the above methods.

The CPCM protocol to enquire the supported geographic location formats of a geographically aware device is shown in figure 27. It consists of the request for the supported geographic location information formats, and the corresponding response.

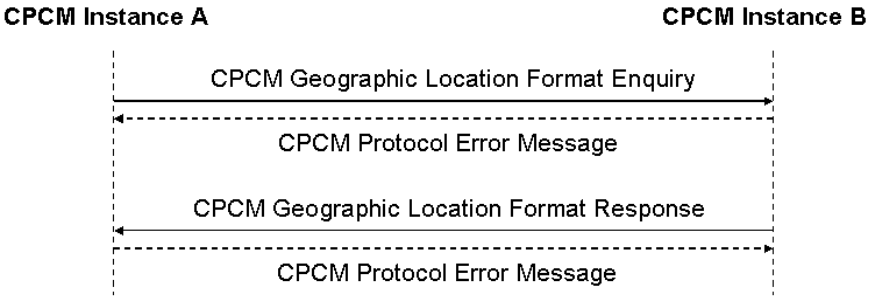

Figure 27: CPCM Geographic Location format enquiry protocol

The CPCM get geographic location protocol is shown in figure 28. It consists of two protocol steps - to request the geographic location in the indicated format, and the corresponding response that provides the geographic location information.

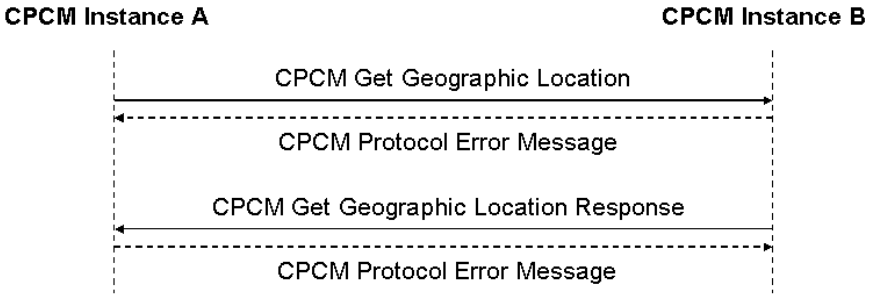

Figure 28: CPCM Geographic information protocol

The CPCM geographic information affirmation protocol is shown in figure 29. It consists of two protocol steps - to request verification that the CPCM Instance is currently located within the provided geographic area, and the corresponding response that either affirms or not presence within that geographic area.

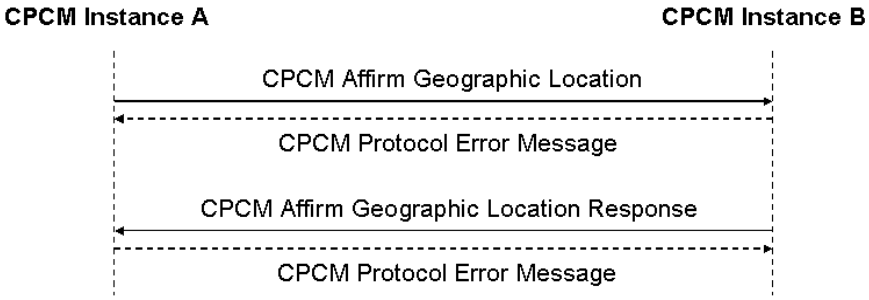

Figure 29: CPCM Geographic Location affirmation protocol

6.5.8.2 CPCM enquire geographic location formats

```
void CPCM_enquire_geo_location_formats_message( void );
```

Table 109 specifies the CPCM protocol message for CPCM enquire Geographic Location formats.

Table 109: CPCM Enquire Geographic Location Formats

Syntax	Bits	Identifier
CPCM_enquire_geo_location_formats_message(){		
CPCM_protocol_message_type = 0x4012	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
}		

Semantics for CPCM_enquire_geo_location_formats_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4012.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 0.

The possible CPCM protocol error codes specific to the CPCM enquire geographic location formats protocol call are listed in table 110.

Table 110: CPCM enquire geographic location formats specific error codes

Error code	Meaning
0x70	CPCM Instance is not geographic aware

6.5.8.3 CPCM geographic location formats response

```
void CPCM_enquire_geo_location_formats_response_message(
    geo_location_format_list                              list
);
```

Table 111 specifies the CPCM protocol message for CPCM geographic location formats response.

Table 111: CPCM Geographic Location Formats response

Syntax	Bits	Identifier
CPCM_enquire_geo_location_formats_response_message(){		
CPCM_protocol_message_type = 0x4013	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
geo_location_formats_list()		CPCM_geo_location_formats_list
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_ enquire_geo_location_formats_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4013.

control_field: This message shall be signed but not encrypted, hence this shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to the number of CPCM_geo_location_format_identiers following this field.

geo_location_formats_list(): This field indicates the CPCM geographic location formats supported by the device, as a series of one-byte identifiers, corresponding to one identifier for each format supported, as defined in The CPCM geographic information structure carries the geographic area to which MGAD or VGAD restriction applies. Its structure is shown in table 18.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8].

There are no protocol-specific error codes defined for this protocol call.

6.5.8.4 CPCM get geographic location

```
void CPCM_get_geo_location_message(
    CPCM_geographic_location_format_code            geo_location_format
);
```

Table 112 specifies the CPCM protocol message for CPCM get geographic location.

Table 112: CPCM Get Geographic Location

Syntax	Bits	Identifier
CPCM_get_geo_location_message(){		
CPCM_protocol_message_type = 0x4010	16	bslbf
control_field = 0X00	8	bslbf
message_payload_length	16	uimsbf
CPCM_geographic_location_format_code	8	uimsbf
}		

Semantics for CPCM_get_geo_location_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4010.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 0.

CPCM_geographic_location_format_code: Format Code in which geographical location information is requested.

The possible CPCM protocol error codes specific to the CPCM get geographic location protocol call are listed in table 113.

Table 113: CPCM get geographic location specific error codes

Error code	Meaning
0x70	CPCM Instance is not geographic aware
0x71	Geographic location format not supported
0x72	Geographic location information currently not available

6.5.8.5 CPCM get geographic location response

```
void CPCM_get_geo_location_response_message(
    CPCM_geographic_location_information            geo_location_information
);
```

Table 114 specifies the CPCM protocol message for CPCM geographic location formats response.

Table 114: CPCM Geographic Location response

Syntax	Bits	Identifier
CPCM_get_geo_location_response_message(){		
CPCM_protocol_message_type = 0x4011	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
geo_location_information()		CPCM_geo_location_information()
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_get_geo_location_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4011.

control_field: This message shall be signed but not encrypted, hence this shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to the number of CPCM_geo_location_format_identiers following this field.

geo_location_information: This carries the requested geographic location information in the requested format.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8].

There are no protocol-specific error codes defined for this protocol call.

6.5.8.6 CPCM affirm geographic location

```
void CPCM_affirm_geo_location_message(
    CPCM_geographic_location_information            geo_location_information
);
```

Table 115 specifies the CPCM protocol message for CPCM affirm geographic location.

Table 115: CPCM Affirm Geographic Location Message

Syntax	Bits	Identifier
CPCM_affirm_geo_location_message(){		
CPCM_protocol_message_type = 0x4014	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
CPCM_geographic_location_information		CPCM_geographic_location_information
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_affirm_geo_location_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4014.

control_field: This message shall be signed but not encrypted, hence this shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

CPCM_geographic_location_information: Geographic area information to be assessed, typically carried within the geographic_area_information field in the CPCM auxiliary data for the Content Item, see clause 5.4.10.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8].

The possible CPCM protocol error codes specific to the CPCM get geographic location protocol call are listed in table 116.

Table 116: CPCM get geographic location specific error codes

Error code	Meaning
0x70	CPCM Instance is not geographic aware
0x71	geographic location format not supported
0x72	geographic location information currently not available

6.5.8.7 CPCM affirm geographic location response

```
void CPCM_affirm_geo_location_response_message(
    CPCM_geo_location_affirmation                   affirmation
);
```

Table 117 specifies the CPCM protocol message for CPCM affirm geographic location response.

Table 117: CPCM Affirm Geographic Location response

Syntax	Bits	Identifier
CPCM_affirm_geo_location_response_message(){		
CPCM_protocol_message_type = 0x4015	16	bslbf
control_field	8	bslbf
message_payload_length	16	uimsbf
geo_location_affirmation	8	bslbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_affirm_geo_location_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4015.

control_field: This message shall be signed but not encrypted, hence this shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 21.

geo_location_affirmation: Code to indicate the affirmation of the passed geographic area. A value of 0x00 indicates that the CPCM Instance is not currently located within the geographic area. A value of 0x01 indicates that the CPCM Instance is currently located within the geographic area.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8].

There are no protocol-specific error codes defined for this protocol call.

6.5.9 AD membership challenge

6.5.9.1 General

The AD membership challenge protocol is used to securely verify the AD membership status of another CPCM Instance prior to the exchange of CPCM Content that is bound to the relevant AD. It consists of two calls, the challenge and associated response, as shown in figure 30.

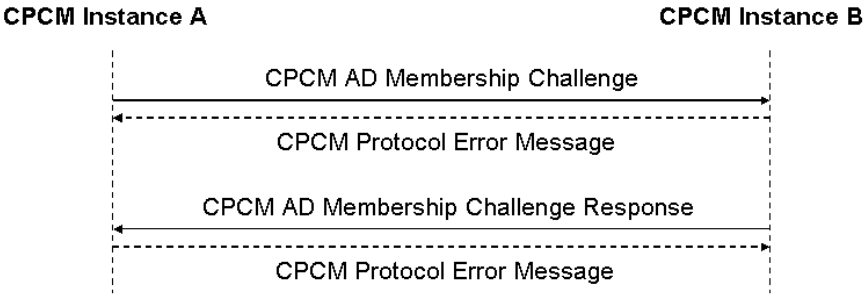

Figure 30: AD membership challenge protocol

The CPCM Instance issuing the challenge communicates the ADID of the AD to which it is a member, with a random nonce value that prevents a replay attack on the message response. If the responding CPCM Instance is indeed a member of the same AD it verifies this by sending the response, which includes the same nonce and the AD Secret signature.

6.5.9.2 CPCM AD membership challenge

```
void CPCM_AD_membership_challenge_message (
    ADID                ADID,
    challenge           random_A
);
```

This message, shown in table 118, shall be sent by the CPCM Instance in order to perform the AD membership challenge.

Table 118: CPCM AD membership challenge message

Syntax	Bits	Identifier
CPCM_AD_membership_challenge_message(){		
CPCM_protocol_message_type = 0x40OE	16	bslbf
control_field = 0xOO	8	bslbf
message_payload_length	16	uimsbf
ADID	72	bslbf
challenge	128	uimsbf
}		

Semantics for CPCM_AD_membership_challenge_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x400E.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 25.

ADID: This is the AD Identifier of the AD of which the challenging CPCM Instance is a member.

challenge: This is a nonce that is chosen by the CPCM Instance issuing the challenge, in order to prevent a replay attack on the message.

The possible CPCM protocol error codes specific to this protocol call are listed in table 119.

Table 119: CPCM AD membership challenge specific error codes

Error code	Meaning
0x80	CPCM Instance is not AD aware
0x81	CPCM Instance is not a member of any AD
0x82	CPCM Instance is not a member of the indicated AD
0x83	The indicated AD has been revoked

6.5.9.3 CPCM AD membership challenge response

```
void CPCM_AD_membership_challenge_response_message (
   challenge            random_A
);
```

This message shall be in response to an AD membership challenge.

Table 120: CPCM AD membership challenge response message

Syntax	Bits	Identifier
ADM_membership_challenge_response_message(){		
CPCM_protocol_message_type = 0x400F	16	bslbf
control_field = 0x04	8	bslbf
message_payload_length	16	uimsbf
challenge	128	uimsbf
AD_secret_signature	160	bslbf
}		

Semantics for ADM_membership_challenge_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x400F.

control_field: This message shall be signed with the AD secret signature but not encrypted, hence this shall be set to 0x04.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 25.

challenge: This shall be set to the same value as the challenge field in the corresponding *CPCM_AD_membership_challenge_message*.

AD_secret_signature: This is the signature of the AD Secret as defined in TS 102 825-5 [i.8], used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 121.

Table 121: CPCM AD membership challenge response specific error codes

Error code	Meaning
0x84	Response challenge mismatch: AD membership not confirmed

6.5.10 CPCM Content Licence Move

6.5.10.1 General

The CPCM Content Licence Move protocol is a transactional CPCM protocol that shall accompany a CPCM Content Move operation, in order to guarantee the integrity of the Move. The CPCM System is agnostic as regards the context of the CPCM Content Move operation, which might depend on the ecosystem or interface at hand.

The same protocol can be applied to either a content pull or content push mode.

Figure 31 shows the Content Licence Move in pull mode, Here nominal CPCM Instance B initiates the CPCM Content Licence Move operation as a result of some action of the user or device application, and pulls the CPCM Content from CPCM Device A. The actual CPCM Content transfer is performed by the home network ecosystem or application-specific physical interface and is independent from the CPCM Content Licence Move protocol sequence.

Content Move can be performed before or after the Content Licence Move and consists in Content Copy followed by a destruction of the original copy.

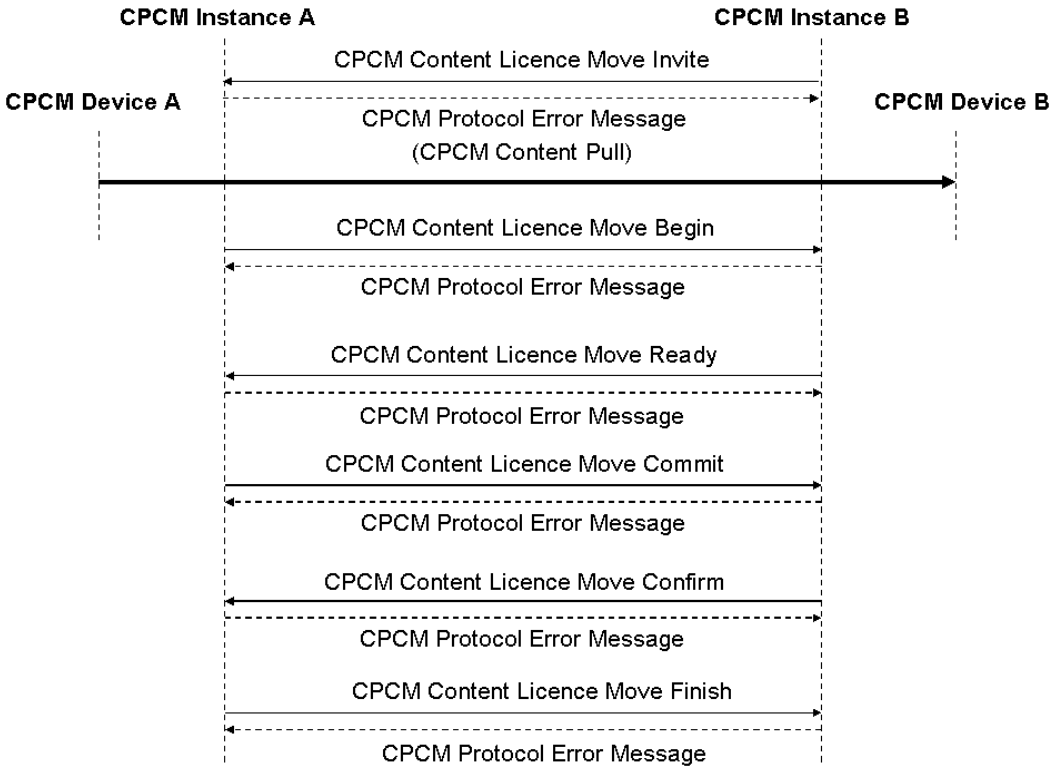

Figure 31: CPCM Content Licence Move protocol, pull mode

Figure 32 shows the Content Licence Move in push mode. Here nominal CPCM Instance A initiates the CPCM Content Licence Move operation as a result of some action of the user or device application, and pushes the CPCM Content Licence to CPCM Device B. The actual CPCM Content transfer is performed by the home network ecosystem or application-specific physical interface and is indepedant from the CPCM Content Licence Move protocol sequence.

Content Move can be performed before or after the Content Licence Move and consists in Content Copy followed by a destruction of the original copy.

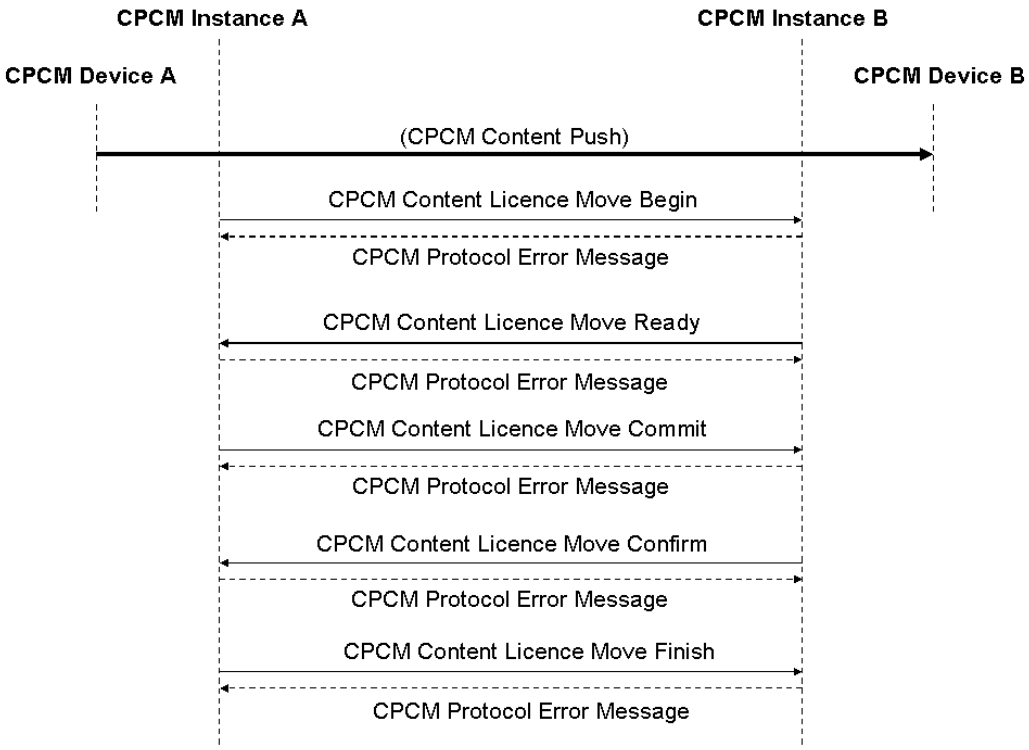

Figure 32: CPCM Content Licence Move protocol, push mode

6.5.10.2 CPCM Content Licence Move begin

```
void CPCM_content_licence_move_begin_message (
    CPCM_content_licence_id                    cl_id,
);
```

Table 122 specifies the CPCM protocol message for CPCM Content Licence Move begin.

Table 122: CPCM Content Licence Move begin

Syntax	Bits	Identifier
CPCM_content_licence_move_begin_message(){		
CPCM_protocol_message_type = 0x4018	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
content_licence_id	128	uimsbf
}		

Semantics for CPCM_content_licence_move_begin_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4018.

control_field: This message shall neither be signed or encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 16.

content_licence_id: CPCM Content Licence Identifier of the CPCM Content item that is to be Moved.

The possible CPCM protocol error codes specific to this protocol call are listed in table 123.

Table 123: CPCM Content Licence Move begin specific error codes

Error code	Meaning
0x20	Unknown Content Licence identifier
0x22	Source CPCM Instance not SE or PE
0x23	Destination CPCM Instance not SE or PE

6.5.10.3 CPCM Content Licence Move ready

```
void CPCM_content_licence_move_ready_message (
    CPCM_content_licence_id                    cl_id,
    CPCM_content_handling_operation            role
);
```

Table 124 specifies the CPCM protocol message for CPCM Content Licence Move ready.

Table 124: CPCM Content Licence Move ready

Syntax	Bits	Identifier
CPCM_content_licence_move_ready_message(){		
CPCM_protocol_message_type = 0x4019	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
content_licence_id	128	uimsbf
role	8	bslbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_content_licence_move_ready_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4019.

control_field: This message shall be signed , hence this shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 37.

content_licence_id: CPCM Content Licence Identifier of the CPCM Content item that is to be Moved.

role: Functional Role of the CPCM Instance replicated from its CPCM Instance Certificate. Its allows the peer CPCM Instance to check the Content Licence will actually be Moved to a Storage or a Processing Entity.

The possible CPCM protocol error codes specific to this protocol call are listed in table 125.

Table 125: CPCM Content Licence Move begin specific error codes

Error code	Meaning
0x21	Content Licence identifier mismatch
0x22	Source CPCM Instance not SE or PE
0x23	Destination CPCM Instance not SE or PE

6.5.10.4 CPCM Content Licence Move commit

```
void CPCM_content_licence_move_commit_message (
    CPCM_content_licence_id                    cl_id,
    CPCM_content_handling_operation            role
);
```

Table 126 specifies the CPCM protocol message for CPCM Content Licence Move commit.

Table 126: CPCM Content Licence Move commit

Syntax	Bits	Identifier
CPCM_content_licence_move_commit_message(){		
CPCM_protocol_message_type = 0x401A	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
content_licence_id	128	uimsbf
role	8	bslbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_content_licence_move_commit_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x401A.

control_field: This message shall be signed , hence this shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 37.

content_licence_id: CPCM Content Licence Identifier of the CPCM Content item that is to be Moved.

role: Functional Role of the CPCM Instance replicated from its CPCM Instance Certificate. Its allows the peer CPCM Instance to check the Content Licence will actually be Moved from a Storage or a Processing Entity.

The possible CPCM protocol error codes specific to this protocol call are listed in table 127.

Table 127: CPCM Content Licence Move begin specific error codes

Error code	Meaning
0x21	Content Licence identifier mismatch
0x22	Source CPCM Instance not SE or PE
0x23	Destination CPCM Instance not SE or PE

6.5.10.5 CPCM Content Licence Move confirm

```
void CPCM_content_licence_move_confirm_message (
    CPCM_content_licence_id                    cl_id
);
```

Table 128 specifies the CPCM protocol message for CPCM Content Licence Move Confirm.

Table 128: CPCM Content Licence Move confirm

Syntax	Bits	Identifier
CPCM_content_licence_move_confirm_message(){		
CPCM_protocol_message_type = 0x401B	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
content_licence_id	128	uimsbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_content_licence_move_confirm_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x401B.

control_field: This message shall be signed , hence this shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 36.

content_licence_id: CPCM Content Licence Identifier of the CPCM Content item that is to be Moved.

The possible CPCM protocol error codes specific to this protocol call are listed in table 129.

Table 129: CPCM Content Licence Move begin specific error codes

Error code	Meaning
0x21	Content Licence identifier mismatch

6.5.10.6 CPCM Content Licence Move finish

```
void CPCM_content_licence_move_finish_message (
    CPCM_content_licence_id                    cl_id
);
```

Table 130 specifies the CPCM protocol message for CPCM Content Licence Move Confirm.

Table 130: CPCM Content Licence Move finish

Syntax	Bits	Identifier
CPCM_content_licence_move_finish_message(){		
CPCM_protocol_message_type = 0x401C	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
content_licence_id	128	uimsbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_content_licence_move_finish_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x401C.

control_field: This message shall be signed , hence this shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 36.

content_licence_id: CPCM Content Licence Identifier of the CPCM Content item that is to be Moved.

The possible CPCM protocol error codes specific to this protocol call are listed in table 131.

Table 131: CPCM Content Licence Move begin specific error codes

Error code	Meaning
0x21	Content Licence identifier mismatch

6.5.10.7 CPCM Content Licence Move request

```
void CPCM_CL_move_request_message (
    CPCM_content_licence_id                        cl_id
);
```

Table 132 specifies the CPCM protocol message for Secure Content Licence Move.

Table 132: CPCM Content Licence Move request

Syntax	Bits	Identifier
CPCM_CL_move_request_message(){		
CPCM_protocol_message_type = 0x401D	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
content_licence_id	128	uimsbf
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_CL_move_request_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x401D.

control_field: This message shall be signed using SAC session key, hence this shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message, hence this shall be set to 36.

content_licence_id: The CPCM Content Licence Identifier of the to be Moved CPCM Content Licence.

The possible CPCM protocol error codes specific to this protocol call are listed in table 133.

Table 133: CPCM Content Licence Move request specific error codes

Error code	Meaning
0x20	Unknown Content Licence identifier
0x22	Source CPCM Instance not SE or PE
0x23	Destination CPCM Instance not SE or PE

6.5.10.8 CPCM Content Licence Move response

```
void CPCM_CL_move_response_message (
    CPCM_content_licence                cl
);
```

This message is used to Move the requested CL. Table 134 specifies the CPCM protocol message for CPCM Content Licence Move response.

Table 134: CPCM Content Licence Move response

Syntax	Bits	Identifier
CPCM_CL_move_response_message(){		
CPCM_protocol_message_type = 0x401E	16	bslbf
control_field	8	bslbf
message_payload_length	16	uimsbf
content_licence		CPCM_content_licence
SAC_secret_signature	160	CPCM_MAC
}		

Semantics for CPCM_CL_move_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x401E.

control_field: This message shall be signed and encrypted using SAC session keys, hence this shall be set to 0x0A.

message_payload_length: The number of message_payload_bytes of the message.

content_licence: The Moved CPCM Content Licence.

SAC_secret_signature: This is the message signature as defined in TS 102 825-5 [i.8].

The possible CPCM protocol error codes specific to this protocol call are listed in table 135.

Table 135: CPCM Content Licence Move response specific error codes

Error code	Meaning
0x21	Content Licence identifier mismatch
0x22	Source CPCM Instance not SE or PE
0x23	Destination CPCM Instance not SE or PE

6.5.11 CPCM Discovery

6.5.11.1 General

The CPCM discovery protocol is a broadcast request and unicast response protocol, as shown in figure 33.

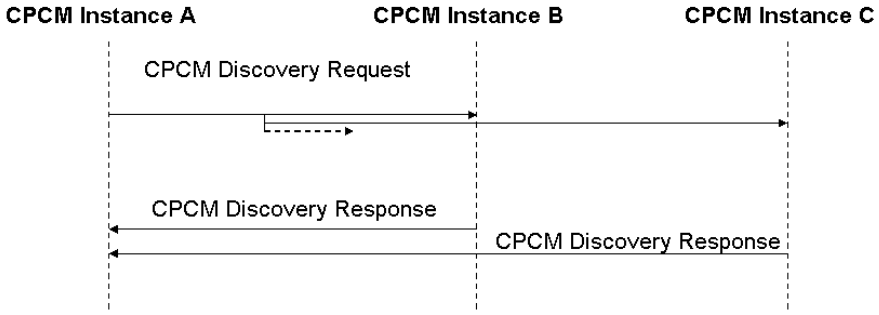

Figure 33: CPCM discovery protocol

6.5.11.2 Discovery request

```
void CPCM_discovery_request_message ( void );
```

This message shall be used to enquire the CPCM-relevant status of other CPCM Instances, including the ADM-relevant status, which will allow, for example, the discovery of which AD(s) are available for a potential new member CPCM Instance to join.

This message is likely to be applied as a broadcast message, depending on the adaptation layer.

For AD-related discovery, the enquiry may be narrowed by including the ADID element as an optional protocol message element, in order to enquire a particular AD.

Table 136: Discovery request Message

Syntax	Bits	Identifier
CPCM_discovery_request_message(){		
CPCM_protocol_message_type = 0x4020	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
[conditional_elements()]		
[optional_elements()]		
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for CPCM_discovery_request_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4020.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message following the message_payload_length field.

There are no protocol-specific error codes defined for this protocol call.

6.5.11.3 Discovery response

```
void CPCM_discovery_response_message (
  CPCM_status                       instance_CPCM_status,
  ADM_status                        instance_ADM_status
);
```

This message shall be used when responding to a *CPCM_Discovery_request_message*,.to provide the general CPCM status and, if applicable, the ADM status of the CPCM Instance.

Table 137: Discovery response Message

Syntax	Bits	Identifier
CPCM_discovery_response_message(){		
CPCM_protocol_message_type = 0x4021	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
instance_CPCM_status	104	CPCM_status
instance_ADM_status	8	ADM_status
[conditional_elements()]		
[optional_elements()]		
}		

NOTE 1: The following elements may be present within the conditional_elements:

- DC_ADSE_values as defined in clause 5.4.19 (present if the Instance is Local Master).
- delegating_instance_id.

NOTE 2: The following elements may be present within the optional_elements:

- AD_name, as defined in clause 5.4.17.

Semantics for CPCM_discovery_response_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4021.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message.

instance_cpcm_status: Structure containing status information about the responding CPCM Instance (see clause 5.4.6).

instance_ADM_status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2. If the CPCM Instance is not AD aware then these bits shall be set to zero.

There are no protocol-specific error codes defined for this protocol call.

6.5.12 CPCM Revocation List Acquisition

6.5.12.1 General

To handle CPCM Content, a CPCM Instance shall have a Revocation List as required in the Content License. If it has not the revocation, it may use the CPCM Revocation List acquisition protocol to know whether some CPCM Instance has it. The Revocation List Transfer itself is dependant upon the underlying network technology and this described in TS 102 825-9 [i.3].

6.5.12.2 Get CPCM Revocation List

```
void get_CPCM_RL_message (
    C_and_R_regime_mask                         mask,
    RL_index_list()                             list
);
```

Table 138 specifies the CPCM protocol message for getting a CPCM revocation list.

Table 138: Get CPCM Revocation List

Syntax	Bits	Identifier
get_CPCM_RL_message(){		
CPCM_protocol_message_type = 0x4016	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
C_and_R_regime_mask	8	bslbf
RL_index_list		
}		

Semantics for get_CPCM_RL_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4016.

control_field: This message shall be neither be signed or encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message.

C_and_R_regime_mask: This mask identifies which Revocation List are requested.

RL_index_list: This list gives for each bit set in the C_and_R_regime_mask in the order from msb to lsb the minimum required revocation list index.

The possible CPCM protocol error codes specific to this protocol call are listed in table 139.

Table 139: Get CPCM Revocation List specific error codes

Error code	Meaning
0x20	CPCM Revocation Lists not available

6.5.12.3 Notify CPCM Revocation List

```
void notify_CPCM_RL_message (
    C_and_R_regime_mask                         mask,
    RL_index_list()                             list
);
```

Table 140 specifies the CPCM protocol message for notifying the availability a CPCM revocation list.

Table 140: Notify CPCM Revocation List

Syntax	Bits	Identifier
notify_CPCM_RL_message(){		
CPCM_protocol_message_type = 0x4017	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
C_and_R_regime_mask	8	bslbf
RL_index_list		
}		

Semantics for notify_CPCM_RL_message:

CPCM_protocol_message_type: According to tables 40 and 70 this shall be set to 0x4017.

control_field: This message shall be neither be signed or encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message.

C_and_R_regime_mask: This mask identifies which Revocation List is available. If none of the requested Revocation Lists are available then this field shall be set to zero.

RL_index_list: This list gives for each bit set in the C_and_R_regime_mask in the order from msb to lsb the available revocation list index.

The possible CPCM protocol error codes specific to this protocol call are listed in table 141.

Table 141: Notify CPCM Revocation List specific error codes

Error code	Meaning
0x21	CPCM Revocation List mismatch

6.6 Authorized Domain management protocols

6.6.1 General

The implementation of this interface and protocol in a CPCM Instance is optional. If the CPCM Instance does not implement ADM then this interface and protocol are not required.

The ADM protocols use some additional ADM-specific enumerated fields, defined in clause 6.6.2.

The CPCM Authorized Domain Management (ADM) Protocols consist of the following functions, specified in subsequent sub-sections:

- AD discovery.
- ADM status enquiry.

Table 142 lists the protocol-specific message codes for Authorized Domain Management (ADM).

As indicated in clause 6.2.2.5, there is a CPCM protocol message category reserved for "ADM relay messages". These are messages that may be defined in a future revision of the present document that are "delegatable". This provides for forwards compatibility in that CPCM Instances that conform with the present document shall forward such delegated messages regardless of whether their content is able to be interpreted by the CPCM Instance.

Table 142: ADM protocol message codes

ADM Message code	Message
0x00	Reserved
0x01	ADM_invite_message
0x02	ADM_invite_result_message
0x03	LM_master_election_request_message
0x04	LM_master_election_response_message
0x05	LM_master_election_indication_message
0x06	AD_update_request_message
0x07	AD_update_response_message
0x08	AD_update_indication_message
0x09	AD_change_request_message
0x0A	AD_change_response_message
0x0B	AD_join_begin_message (unicast)
0x0C	AD_join_begin_message (broadcast)
0x0D	AD_join_ready_message
0x0E	AD_join_commit_message
0x0F	AD_join_confirm_message
0x10	AD_join_finish_message
0x11	AD_leave_begin_message
0x12	AD_leave_ready_message
0x13	AD_leave_commit_message
0x14	AD_leave_confirm_message
0x15	AD_leave_finish_message
0x16	DC_transfer_begin_message
0x17	DC_transfer_ready_message
0x18	DC_transfer_commit_message
0x19	DC_transfer_confirm_message
0x1A	DC_transfer_finish_message
0x1B	DC_split_begin_message
0x1C	DC_split_ready_message
0x1D	DC_split_commit_message
0x1E	DC_split_confirm_message
0x1F	DC_split_finish_message
0x20	DC_merge_begin_message
0x21	DC_merge_ready_message
0x22	DC_merge_commit_message
0x23	DC_merge_confirm_message
0x24	DC_merge_finish_message
0x25	DC_rebalance_begin_message
0x26	DC_rebalance_ready_message
0x27	DC_rebalance_commit_message
0x28	DC_rebalance_confirm_message
0x29	DC_rebalance_finish_message
0x2A to 0x7F	reserved
0x80	become_DC_message
0x81	new_DC_message
0x82	merge_DC_message
0x83	DC_merged_message
0x84	rebalance_DC_message
0x85	DC_rebalanced_message
0x86	ADM_quorum_test_query_message
0x87	ADM_quorum_test_response_message
0x88 to 0xEF	reserved
0xF0	ADMAAA_tool_request_message
0xF1	ADMAAA_tool_response_message
0xF2 to 0xFF	reserved

Table 143 provides the complete list of error codes and their meanings used defined for use in the CPCM ADM protocols. Not all errors are valid for each individual protocol call. Since the set of ADM protocols has a well-defined scope there are no individual lists of possible error messages defined for each ADM protocol call.

Table 143: Summary of ADM specific error codes

Error code	Meaning
0x20 to 0x3F	Reserved for future use
0x40	No Domain Controller is available
0x41	Requested Domain Controller is not available
0x42	Instance is not a Domain Controller
0x43	Instance is not a Local Master
0x44	Instance is neither a Domain Controller nor a Local Master
0x45	Remote_ceiling exceeded
0x46	Local_ceiling exceeded and negative quorum test
0x47	Total_ceiling exceeded
0x48	DC_remote_ceiling exceeded
0x49	DC_ceiling exceeded
0x4A	History_ceiling exceeded
0x4B	Rejection by ADMAAA
0x4C	Challenge mismatch in quorum test response
0x4D	Bad ADSE counts requested
0x4E to 0x4F	Reserved for future use
0x50	Inconsistent internal_record_list
0x51	Outdated internal_record_list
0x52	Internal_record with bad signature
0x53	Outdated internal_record
0x54	Renaming refused
0x55	Name mismatch
0x56 to 0x7F	Reserved for future use
0x80	CPCM Instance is not AD aware
0x81	CPCM Instance is not a member of any AD
0x82	CPCM Instance is not a member of the indicated AD
0x83	CPCM Instance is not ADM capable
0x84	Instance is not Local Master capable
0x85	Instance is not Domain Controller capable
0x86	Instance is not ADSE capable
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x89	Unexpected ADSE values
08A to 0x8F	Reserved for future use
0x90	Instance is already AD member
0x91	Certificate-message mismatch
0x92	LM capability too low
0x93	Too many Local Masters
0x94 to 0xFF	Reserved for future use

6.6.2 ADM Enumerated Fields

6.6.2.1 General

This clause defines the enumerated fields used in the ADM part of the CPCM protocols, including the ADM-specific error codes.

6.6.2.2 ADM status

The ADM_status field indicates the current status of the CPCM Instance as regards its role within an AD, if it is a member of one. The possible states are listed in table 144.

Table 144: ADM status

Value	ADM status
0	Blank
1	AD member
2	Local Master
3	Domain Controller
4	Local Master and Domain Controller

6.6.2.3 ADM condition

This field is carried when using the ADMAAA tool. It describes the circumstances under which ADMAAA intervention is necessary. The possible states are listed in table 145.

Table 145: AD condition

Value	AD Condition
0	Total_ceiling exceeded
1	Local_ceiling exceeded
2	Remote_ceiling exceeded
3	Quorum test failure
4	DC_ceiling exceeded
5	DC_remote_ceiling exceeded

6.6.2.4 ADM protocol

This field is carried upon invitation to start an AD protocol.

Table 146: AD protocol

Value	Meaning
0	Join AD
1	Leave AD
2	Domain Controller Transfer
3	Domain Controller Split
4	Domain Controller Merge
5	Domain Controller Rebalance

6.6.2.5 Delegating CPCM Instance identifier

The presence of the delegating_instance_id field indicates that a delegated action is being performed. This means the message is exchanged either between a Local Master and a Domain Controller, the protocol being run at the request of a third party or from the Local Master to the third party, at the request of the Domain Controller. The field carries either the CIC Identifier of the requesting CPCM Instance, or the CIC Identifier of the Domain Controller when the Local Master forwards its response, depending on the context of the message.

6.6.2.6 Local Master capability

This field shall carry a value for use in electing the most capable CPCM Instance as the new Local Master. The process of electing a new Local Master is defined in the ADM specification (TS 102 825-7 [i.9]). The LM_capability field shall assume one of the values defined in table 147.

Table 147: LM Capability

Value	LM Capability
0	Instance is LM capable but does not want to take Local Master Role
1	Local Master capable Instance
2	Domain Controller capable Instance
3	Current Domain Controller
4	Instance needs to get the Local Master Role

6.6.2.7 CPCM version

This field carries the version of the implemented CPCM specification. The CPCM_version shall be 0x01 for this first version of the present document. It is optional in all messages. If absent, it is assumed to be version 1.

6.6.3 General ADM protocols

6.6.3.1 General

This clause specifies the collection of general ADM protocols, consisting of the following components:

- AD update, for a CPCM Instance to request update indications of the AD status.
- AD change; for a CPCM Instance to request a Domain Controller to change the registered AD attributes.
- ADM quorum test, for a Domain Controller to establish whether the conditions to allow a new CPCM Instance to join the AD are satisfied, as specified in clause 6.6.3.7.
- ADM invite, for when one CPCM Instance invites another CPCM Instance to initiate an ADM protocol.

6.6.3.2 AD update request

```
void AD_update_request_message (
  AD_internal_record_list()     internal_record_list
);
```

This message shall be broadcast by a member Instance that wants to be notified of the current state of the Domain, e.g. after reconnection to the network.

Table 148: AD update request message

Syntax	Bits	Identifier
AD_update_request_message(){		
CPCM_protocol_message_type = 0x6006	16	bslbf
control_field = 0x04	8	bslbf
message_payload_length	16	uimsbf
internal_record_list		AD_internal_record_list()
[optional_elements()]		
AD_secret_signature	160	Bslbf
}		

Semantics for AD_update_request_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6006.

control_field: This message shall be protected by the AD secret, hence this field shall be set to 0x04.

message_payload_length: The number of message_payload_bytes of the message.

internal_record_list(): The list of internal records.

AD_secret_signature: The signature of the AD Secret as defined in TS 102 825-5 [i.8], used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 149.

Table 149: AD update request specific error codes

Error code	Meaning
0x44	Instance is neither a Domain Controller nor a Local Master
0x50	Inconsistent internal_record_list
0x52	Internal_record with bad signature
0x88	The indicated AD has been revoked

6.6.3.3 AD update response

```
void AD_update_response_message ( void );
```

This message shall be sent by the Local Master in response to an AD Update request.

Table 150: AD update response message

Syntax	Bits	Identifier
AD_update_response_message(){		
CPCM_protocol_message_type = 0x6007	16	bslbf
control_field = 0x04	8	bslbf
message_payload_length	16	uimsbf
[conditional_elements()]		
[optional_elements()]		
AD_secret_signature	160	bslbf
}		

Semantics for AD_update_response_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6007.

control_field: This message shall be protected by the AD Secret, hence this field shall be set to 0x04.

message_payload_length: The number of message_payload_bytes of the message.

conditional_elements: This shall include the element internal_record_list for those internal records that have been updated, if any.

AD_secret_signature: This is the signature of the AD Secret as defined in TS 102 825-5 [i.8], used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 151.

Table 151: AD update response specific error codes

Error code	Meaning
0x50	Inconsistent internal_record_list
0x51	Outdated internal_record_list
0x52	Internal_record with bad signature
0x87	ADID mismatch
0x88	The indicated AD has been revoked

6.6.3.4 AD update indication

```
void AD_update_indication_message (
  AD_internal_record_list()     internal_record_list
);
```

This message shall be broadcast by the Domain Controller and/or Local Master whenever a change occurs to the Domain Internal Record.

Table 152: AD update indication message

Syntax	Bits	Identifier
AD_update_indication_message(){		
CPCM_protocol_message_type = 0x6008	16	bslbf
control_field = 0x04	8	bslbf
message_payload_length	16	uimsbf
internal_record_list		AD_internal_record_list()
[optional_elements()]		
AD_secret_signature	160	bslbf
}		

NOTE: Only Domain Internal Records that have just been updated need to be incorporated in internal_record_list.

Semantics for AD_update_indication_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6008.

control_field: This message shall be protected by the AD Secret, hence this field shall be set to 0x04.

message_payload_length: The number of message_payload_bytes of the message.

internal_record_list(): The list of internal records.

AD_secret_signature: The signature of the AD Secret as defined in TS 102 825-5 [i.8], used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in Table 153.

Table 153 : AD update indication specific error codes

Error code	Meaning
0x50	Inconsistent internal_record_list
0x51	Outdated internal_record_list
0x52	Internal_record with bad signature
0x87	ADID mismatch
0x88	The indicated AD has been revoked

6.6.3.5 AD change request

```
void AD_change_request_message (
  ADID                  ADID,
  AD_name               name
);
```

This message shall be sent to the Domain Controller (possibly via Local Master) by a non-Domain Controller Instance in order to request a change to the Domain.

Table 154: AD change request message

Syntax	Bits	Identifier
AD_change_request_message(){		
CPCM_protocol_message_type = 0x6009	16	bslbf
control_field = 0x04	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
new_AD_name_requested	see clause 5.4.17	AD_name()
[conditional_elements()]		
[optional_elements()]		
AD_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for AD_change_request_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6009.

control_field: This message shall be protected by the AD Secret, hence this field shall be set to 0x04.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD requesting the change.

new_AD_name_requested: The new name requested for the AD.

AD_secret_signature: The signature of the AD Secret as defined in TS 102 825-5 [i.8], used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 155.

Table 155: AD change request specific error codes

Error code	Meaning
0x40	No Domain Controller is available
0x41	Requested Domain Controller is not available
0x44	Instance is neither a Domain Controller nor a Local Master
0x54	Renaming refused
0x87	ADID mismatch
0x88	The indicated AD has been revoked

6.6.3.6 AD change response

```
void AD_change_response_message (
  ADID                  ADID,
  AD_name               name
);
```

This message shall be the response from the Domain Controller when a change has been requested.

Table 156: AD change response message

Syntax	Bits	Identifier
AD_change_response_message(){		
CPCM_protocol_message_type = 0x600A	16	bslbf
control_field = 0x04	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
new_AD_name	see clause 5.4.17	AD_name()
[conditional_elements()]		
[optional_elements()]		
AD_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for AD_change_ response_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x600A.

control_field: This message shall be protected by the AD Secret, hence this field shall be set to 0x04.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD requesting the change.

new_AD_name(): The new name of the AD.

AD_secret_signature: The signature of the AD Secret as defined in TS 102 825-5 [i.8], used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 157.

Table 157: AD change response specific error codes

Error code	Meaning
0x55	Name mismatch
0x87	ADID mismatch
0x88	The indicated AD has been revoked

6.6.3.7 ADM quorum test query

```
void ADM_quorum_test_query_message (
  ADID                  ADID,
  challenge             Random_A
);
```

This message shall be sent by the Domain Controller to perform the quorum test during the Join protocol.

Table 158: Quorum test query message

Syntax	Bits	Identifier
ADM_quorum_test_query_message(){		
CPCM_protocol_message_type = 0x6086	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
challenge	128	challenge
[optional_elements()]		
}		

Semantics for ADM_quorum_test_query_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6086.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD requesting the test.

challenge: This is a nonce that is chosen by the CPCM Instance issuing the challenge, in order to prevent a replay attack on the message.

The possible CPCM protocol error codes specific to this protocol call are listed in table 159.

Table 159: AD quorum test query specific error codes

Error code	Meaning
0x87	ADID mismatch
0x88	The indicated AD has been revoked

6.6.3.8 ADM quorum test response

```
void ADM_quorum_test_response_message (
  CPCM_instance_certificate_id                originating_instance_id,
  AD_capability()                             capability,
  challenge                                   Random_A
);
```

This message shall be in response to a quorum test query from the same AD.

Table 160: Quorum test response message

Syntax	Bits	Identifier
ADM_quorum_test_response_message(){		
CPCM_protocol_message_type = 0x6087	16	bslbf
control_field = 0x04	8	bslbf
message_payload_length	16	uimsbf
originating_instance_id	64	CPCM_instance_certificate_id
capability	8	AD_capability()
challenge	128	uimsbf
[optional_elements()]		
AD_secret_signature	160	bslbf
}		

Semantics for ADM_quorum_test_response_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6087.

control_field: This message shall be protected by the AD Secret, hence this field shall be set to 0x04.

message_payload_length: The number of message_payload_bytes of the message.

originating_instance_id: The CPCM Instance sending the response shall set this field to its CPCM Instance identifier, in order to guarantee the uniqueness of each quorum test response message received by the quorum tester.

capability: Carries the replication of the five AD and ADM capability fields of the CPCM Instance Certificate.

challenge: This shall be set to the same value as the challenge field in the corresponding *ADM_quorum_test_query_message*.

AD_secret_signature: The signature of the AD Secret as defined in TS 102 825-5 [i.8], used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 161.

Table 161: AD quorum test response specific error codes

Error code	Meaning
0x4C	Challenge mismatch in quorum test response
0x86	Instance is not ADSE capable
0x88	The indicated AD has been revoked

6.6.3.9 ADM invite

```
void ADM_invite_message (
  ADID                                    ADID,
  CPCM_protocol                           protocol,
  CPCM_instance_certificate_id            domain_controller
);
```

This message is used when a Device Application requests an Instance to start an ADM security protocol from another networked Instance.

Table 162: ADM invite message

Syntax	Bits	Identifier
ADM_invite_message(){		
CPCM_protocol_message_type = 0x6001	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
protocol	8	CPCM_protocol
[domain_controller]	64	CPCM_instance_certificate_id
[ADSE_values]		ADSE_values()
[optional_elements()]		
}		

Semantics for ADM_invite_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6001.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

protocol: Carries a value for the type of transaction, see table 146.

domain_controller: This field is optional. If present it indicates the ID of the specific Domain Controller being addressed.

ADSE_values: This field is conditional. It is applicable for the DC split and DC rebalance protocols.

The possible CPCM protocol error codes specific to this protocol call are listed in table 163.

Table 163: ADM invite specific error codes

Error code	Meaning
0x40	No Domain Controller is available
0x41	Requested Domain Controller is not available
0x42	Instance is not a Domain Controller
0x4A	History_ceiling exceeded
0x80	CPCM Instance is not AD aware
0x81	CPCM Instance is not a member of any AD
0x82	CPCM Instance is not a member of the indicated AD
0x83	CPCM Instance is not ADM capable
0x85	Instance is not Domain Controller capable
0x87	ADID mismatch
0x88	The indicated AD has been revoked

6.6.3.10 ADM invite result

```
void ADM_invite_result_message (
  ADID                                ADID,
  CPCM_protocol                       protocol,
  CPCM_instance_certificate_id        domain_controller
);
```

This message shall be the response to *ADM_invite_message*.

Table 164: ADM invite result message

Syntax	Bits	Identifier
ADM_invite_result_message(){		
CPCM_protocol_message_type = 0x6002	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
protocol	8	CPCM_protocol
domain_controller	64	bslbf
[optional_elements()]		
}		

Semantics for ADM_invite_result_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6002.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

protocol: Carries a value for the type of transaction, see table 146.

domain_controller: The ID of the Domain Controller.

The possible CPCM protocol error codes specific to this protocol call are listed in table 165.

Table 165: ADM invite result specific error codes

Error code	Meaning
0x87	ADID mismatch
0x88	The indicated AD has been revoked

6.6.4 AD Join

6.6.4.1 General

The AD Join protocol is a transactional protocol used to enable a CPCM Instance to Join an existing AD.

6.6.4.2 AD Join begin

```
void AD_join_begin_message (
  ADID                                ADID,
  ADM_status                          status,
  AD_capability()                     capability
);
```

This message shall be used by a CPCM Instance to initiate Joining an existing AD. This message can be sent (message type is 0x600C) or broadcast (message type is 0x600D).

Table 166: AD Join begin message

Syntax	Bits	Identifier
AD_join_begin_message(){		
CPCM_protocol_message_type = 0x600B || 0x600C	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
status	8	ADM_status
capability	8	AD_capability()
[conditional_elements()]		
[optional_elements()]		
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for AD_join_begin_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x600B if sent or 0x600C if broadcast.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

capability: Carries the replication of the five AD and ADM capability fields of the CPCM Instance Certificate.

The possible CPCM protocol error codes specific to this protocol call are listed in table 167.

Table 167: AD join begin specific error codes

Error code	Meaning
0x40	No Domain Controller is available
0x41	Requested Domain Controller is not available
0x42	Instance is not a Domain Controller
0x43	Instance is not a Local Master
0x45	Remote_ceiling exceeded
0x46	Local_ceiling exceeded and negative quorum test
0x47	Total_ceiling exceeded
0x4B	Rejection by ADMAAA
0x80	CPCM Instance is not AD aware
0x81	CPCM Instance is not a member of any AD
0x82	CPCM Instance is not a member of the indicated AD
0x83	CPCM Instance is not ADM capable
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x90	Instance is already AD member
0x91	Certificate-message mismatch

6.6.4.3 AD Join ready

```
void AD_join_ready_message (
  ADM_status                        status,
);
```

This message shall be used by a Local Master and/or Domain Controller to begin acceptance of a Join process.

Table 168: AD Join ready message

Syntax	Bits	Identifier
AD_join_ready_message(){		
CPCM_protocol_message_type = 0x600D	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
status	8	ADM_status
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for AD_join_ready_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x600D.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

SAC_secret_signature: This is the message signature using SAC session key.

The possible CPCM protocol error codes specific to this protocol call are listed in table 169.

Table 169 : AD join ready specific error codes

Error code	Meaning
0x42	Instance is not a Domain Controller
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x91	Certificate-message mismatch

6.6.4.4 AD Join commit

```
void AD_join_commit_message ( void );
```

This message shall be used to indicate full commitment by the Instance wanting to Join the Domain.

Table 170: AD Join commit message

Syntax	Bits	Identifier
AD_join_commit_message(){		
CPCM_protocol_message_type = 0x600E	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

NOTE 1: After this is sent, the originator of the request has no further opportunity to cancel the Join process.

NOTE 2: The delegating_instance_id element may be present within the conditional_elements.

Semantics for AD_join_commit_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x600E.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using SAC session key.

There are no protocol-specific error codes defined for this protocol call.

6.6.4.5 AD Join confirm

```
void AD_join_confirm_message ( void );
```

This message shall be sent by the Domain Controller and/or Local Master to the new Instance to confirm a Join.

Table 171: AD Join confirm message

Syntax	Bits	Identifier
AD_join_confirm_message(){		
CPCM_protocol_message_type = 0x600F	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for AD_join_confirm_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x600F.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using SAC session key.

There are no protocol-specific error codes defined for this protocol call.

6.6.4.6 AD Join finish

```
void AD_join_finish_message ( void );
```

This message shall be sent back by the newly Joined Instance to allow cleanup of Join records at the Local Master and Domain Controller.

Table 172: AD Join Finish message

Syntax	Bits	Identifier
AD_join_finish_message(){		
CPCM_protocol_message_type = 0x6010	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for AD_join_finish_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6010.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using SAC session key.

There are no protocol-specific error codes defined for this protocol call.

6.6.5 AD Leave

6.6.5.1 General

The AD Leave protocol is a transactional protocol used to enable a CPCM Instance to leave the AD of which it is a member.

6.6.5.2 AD Leave begin

```
void AD_leave_begin_message (
  ADID                                ADID,
  ADM_status                          status,
  AD_capability()                     capability
);
```

This message shall be sent by any member Instance when it wishes to remove itself from the Domain.

Table 173 : AD Leave begin message

Syntax	Bits	Identifier
AD_leave_begin_message(){		
CPCM_protocol_message_type = 0x6011	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
ADID	72	bslbf
status	8	ADM_status
capability	8	AD_capability()
[delegating_instance_id]	64	CPCM_instance_certificate_id
[optional_elements()]		
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for AD_leave_begin_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6011.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

capability(): Carries the replication of the five AD and ADM capability fields of the CPCM Instance Certificate.

The possible CPCM protocol error codes specific to this protocol call are listed in table 174.

Table 174: AD leave begin specific error codes

Error code	Meaning
0x40	No Domain Controller is available
0x41	Requested Domain Controller is not available
0x42	Instance is not a Domain Controller
0x43	Instance is not a Local Master
0x80	CPCM Instance is not AD aware
0x81	CPCM Instance is not a member of any AD
0x82	CPCM Instance is not a member of the indicated AD
0x83	CPCM Instance is not ADM capable
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x91	Certificate-message mismatch

6.6.5.3 AD Leave ready

```
void AD_leave_ready_message (
  ADID                          ADID,
  ADM_status                    status
);
```

This message shall be sent by the Local Master or Domain Controller in response to the Instance wishing to Leave.

Table 175: AD Leave ready message

Syntax	Bits	Identifier
AD_leave_ready_message(){		
CPCM_protocol_message_type = 0x6012	16	bslbf
control_field = 0x0C	8	bslbf
payload_length	16	uimsbf
ADID	72	ADID
status	8	ADM_status
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
AD_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for AD_leave_ready_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6012.

control_field: This message shall be signed by both the SAC session key and the AD secret, hence this field shall be set to 0x0C.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

SAC_secret_signature: This is the message signature using SAC session key.

AD_secret_signature: The signature of the AD Secret as defined in TS 102 825-5 [i.8], used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 176.

Table 176 : AD leave ready specific error codes

Error code	Meaning
0x42	Instance is not a Domain Controller
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x91	Certificate-message mismatch

6.6.5.4 AD Leave commit

```
void AD_leave_commit_message (
  ADID                              ADID,
  ADM_status                        status
);
```

This message shall be sent to the Local Master or Domain Controller by the Instance wishing to Leave. The originator shall have no further chance to cancel the operation.

Table 177: AD Leave commit message

Syntax	Bits	Identifier
AD_leave_commit_message(){		
CPCM_protocol_message_type = 0x6013	16	bslbf
control_field = 0x0C	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
status	8	ADM_status
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
AD_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements element.

Semantics for AD_leave_commit_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6013.

control_field: This message shall be signed by both the SAC session key and the AD secret, hence this field shall be set to 0x0C.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

SAC_secret_signature: This is the message signature using SAC session key.

AD_secret_signature: The signature of the AD Secret as defined in TS 102 825-5 [i.8], used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 178.

Table 178: AD leave commit specific error codes

Error code	Meaning
0x81	CPCM Instance is not a member of any AD
0x82	CPCM Instance is not a member of the indicated AD
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x91	Certificate-message mismatch

6.6.5.5 AD Leave confirm

```
void AD_leave_confirm_message ( void );
```

This message shall provide the confirmation that a Leave operation has been completed.

Table 179: AD Leave confirm message

Syntax	Bits	Identifier
AD_leave_confirm_message(){		
CPCM_protocol_message_type = 0x6014	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for AD_leave_confirm_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6014.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using SAC session key.

There are no protocol-specific error codes defined for this protocol call.

6.6.5.6 AD Leave finish

```
void AD_leave_finish_message ( void );
```

This message shall be the final message from the departing Instance to the Domain, allowing cleanup of change records.

Table 180: AD Leave finish message

Syntax	Bits	Identifier
AD_leave_finish_message(){		
CPCM_protocol_message_type = 0x6015	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for AD_leave_finish_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6015.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using SAC session key.

There are no protocol-specific error codes defined for this protocol call.

6.6.6 Local Master election

6.6.6.1 General

The Local Master (LM) election protocol is applied in order to select the CPCM Instance that is to perform the role of ADM Local Master when more than one CPCM Instance is present that could assume the role of Local Master. This process is defined in detail in TS 102 825-7 [i.9].

The protocol sequence is shown in figure 34 as an example scenario where CPCM Instance A initiates the LM election protocol, CPCM Instances B and C respond, and CPCM Instance B wins the election, based on the rules as defined in the ADM Specification (TS 102 825-7 [i.9]).

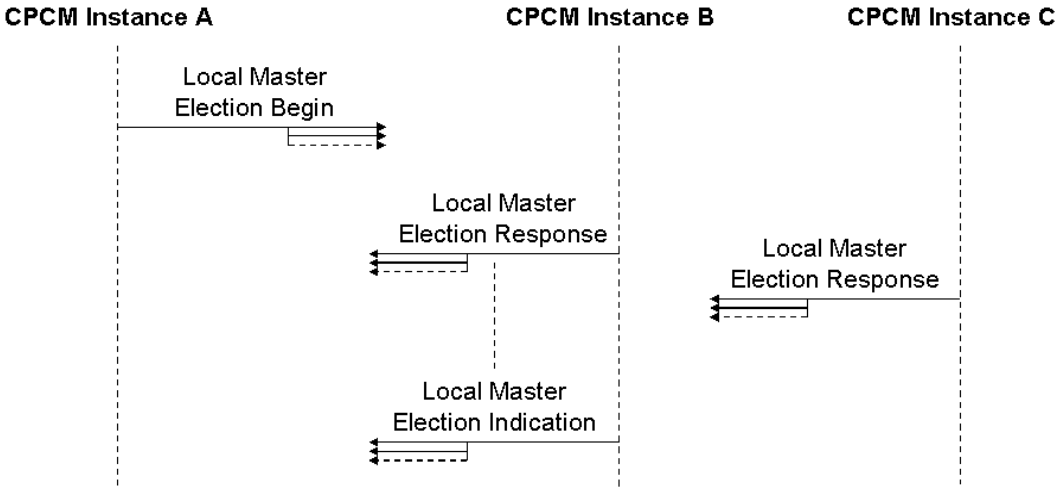

Figure 34: ADM Local Master election protocol

The individual protocol steps are specified in the following clauses.

6.6.6.2 Local Master election request

```
void LM_ election_request_message (
  ADM_status                        status,
  ADID                              ADID
);
```

This message shall be broadcast by any member Instance on the local network to request the election of a new Local Master.

Table 181: Local Master election request message

Syntax	Bits	Identifier
LM_election_request_message(){		
CPCM_protocol_message_type = 0x6003	16	bslbf
control_field = 0x04	8	bslbf
message_payload_length	16	uimsbf
status	8	ADM_status
ADID	72	ADID
[conditional_elements()]		
[optional_elements()]		
AD_secret_signature	160	bslbf
}		

NOTE: The following elements may be present within the conditional_elements:

- ADSE_values() is the requesting CPCM Instance is a Domain Controller.
- LM_capability, as defined in clause 6.6.2.6, if the requesting CPCM Instance is Local Master capable.

Semantics for LM_ election_request_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6003.

control_field: This message shall be signed by the AD secret, hence this field shall be set to 0x04.

message_payload_length: The number of message_payload_bytes of the message.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

ADID: The AD Identifier of the AD.

AD_secret_signature: The signature of the AD Secret, used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 182.

Table 182: LM election request specific error codes

Error code	Meaning
0x88	The indicated AD has been revoked

6.6.6.3 Local Master election response

```
void LM_ election_response_message (
  ADID                                ADID,
  ADM_status                          status,
  LM_capability                       Local_Master_capability
);
```

This message shall be broadcast by each member Instance in response to a Local Master Election request.

Table 183: Local Master election response message

Syntax	Bits	Identifier
LM_ election_response_message(){		
CPCM_protocol_message_type = 0x6004	16	bslbf
control_field = 0x04	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
status	8	ADM_status
local_master_capability	8	LM_capability
[conditional_elements()]		
[optional_elements()]		
AD_secret_signature	160	bslbf
}		

Semantics for LM_ election_response_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6004.

control_field: This message shall be signed by the AD secret, hence this field shall be set to 0x04.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

local_master_capability: See table 147.

conditional_elements: If the CPCM Instance issuing the message is a Domain Controller, its current ADSE_values element shall be included here.

AD_secret_signature: The signature of the AD Secret, used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 184.

Table 184: LM election response specific error codes

Error code	Meaning
0x84	Instance is not Local Master capable
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x92	LM capability too low

6.6.6.4 Local Master election indication

```
void LM_ election_indication_message (
  ADID                              ADID
);
```

This message shall be broadcast by the winner of a Local Master election.

Table 185: Local Master Election Indication message

Syntax	Bits	Identifier
LM_ election_indication_message(){		
CPCM_protocol_message_type = 0x6005	16	bslbf
control_field = 0x04	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
[optional_elements()]		
AD_secret_signature	160	bslbf
}		

Semantics for LM_ election_ indication_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6005.

control_field: This message shall be signed by the AD secret, hence this field shall be set to 0x04.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

AD_secret_signature: The signature of the AD Secret, used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 186.

Table 186: LM election indication specific error codes

Error code	Meaning
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x93	Too many Local Masters

6.6.7 Domain Controller transfer

6.6.7.1 General

The Domain Controller transfer protocol is a transactional protocol applied in order to transfer the role of Domain Controller to another CPCM Instance, as defined in TS 102 825-7 [i.9].

6.6.7.2 Domain Controller transfer begin

```
void DC_transfer_begin_message (
  ADID                                ADID,
  ADM_status                          status,
  AD_capability()                     capability
);
```

This message shall be sent to a Domain Controller to signal the start of a Domain Controller transfer operation.

Table 187: Domain Controller Transfer begin message

Syntax	Bits	Identifier
DC_transfer_begin_message(){		
CPCM_protocol_message_type = 0x6016	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
status	8	ADM_status
capability	64	AD_capability()
[conditional_elements()]		
[optional_elements()]		
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_transfer_begin_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6016.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

capability(): Carries the replication of the five AD and ADM capability fields of the CPCM Instance Certificate.

The possible CPCM protocol error codes specific to this protocol call are listed in table 188.

Table 188: DC transfer begin specific error codes

Error code	Meaning
0x40	No Domain Controller is available
0x41	Requested Domain Controller is not available
0x42	Instance is not a Domain Controller
0x48	DC_remote_ceiling exceeded
0x4B	Rejection by ADMAAA
0x80	CPCM Instance is not AD aware
0x81	CPCM Instance is not a member of any AD
0x82	CPCM Instance is not a member of the indicated AD
0x83	CPCM Instance is not ADM capable
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x91	Certificate-message mismatch

6.6.7.3 Domain Controller transfer ready

```
void DC_transfer_ready_message (
  ADID                                ADID,
  ADM_status                          status,
  ADSE_values()                       ADSE_values
);
```

This message shall be returned by a Domain Controller when prepared for a Domain Controller transfer operation.

Table 189: Domain Controller transfer ready message

Syntax	Bits	Identifier
DC_transfer_ready_message(){		
CPCM_protocol_message_type = 0x6017	16	bslbf
control_field = 0x0C	8	bslbf
message_payload_length	16	uimsbf
ADID	72	bslbf
status	8	ADM_status
ADSE_values		ADSE_values()
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
AD_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_transfer_ready_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6017.

control_field: This message shall be signed by both the SAC session key and the AD secret, hence this field shall be set to 0x0C.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

ADSE_values: Carries the current state of the ADSE variables associated with an AD (see clause 5.4.19).

SAC_secret_signature: This is the message signature using SAC session key.

AD_secret_signature: The signature of the AD Secret, used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 190.

Table 190: DC transfer ready specific error codes

Error code	Meaning
0x42	Instance is not a Domain Controller
0x80	CPCM Instance is not AD aware
0x83	CPCM Instance is not ADM capable
0x85	Instance is not Domain Controller capable
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x89	Unexpected ADSE values
0x91	Certificate-message mismatch

6.6.7.4 Domain Controller transfer commit

```
void DC_transfer_commit_message ( void );
```

This message shall be sent to a Domain Controller to proceed to a Domain Controller transfer operation.

Table 191: Domain Controller transfer commit message

Syntax	Bits	Identifier
DC_transfer_commit_message(){		
CPCM_protocol_message_type = 0x6018	16	bslbf
control_field = 0x0C	8	bslbf
message_payload_length	16	uimsbf
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
AD_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_transfer_commit_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6018.

control_field: This message shall be signed by both the SAC session key and the AD secret, hence this field shall be set to 0x0C.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using SAC session key.

AD_secret_signature: The signature of the AD Secret, used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 192.

Table 192: DC transfer commit specific error codes

Error code	Meaning
0x42	Instance is not a Domain Controller
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x91	Certificate-message mismatch

6.6.7.5 Domain Controller transfer confirm

```
void DC_transfer_confirm_message (
  AD_internal_record        internal_record
);
```

This message shall provide the confirmation that Domain Controller transfer has been completed.

Table 193: Domain Controller transfer confirm message

Syntax	Bits	Identifier
DC_transfer_confirm_message(){		
CPCM_protocol_message_type = 0x6019	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
internal_record		AD_internal_record
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

NOTE 1: The internal_record embeds the updated Domain Internal Record (i.e. with updated Domain controller identifier and updated record index).

NOTE 2: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_transfer_confirm_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6019.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

internal_record(): The internal record for the DC. See clause 5.4.22.

SAC_secret_signature: This is the message signature using SAC session key.

The possible CPCM protocol error codes specific to this protocol call are listed in table 194.

Table 194 : DC transfer confirm specific error codes

Error code	Meaning
0x52	Internal_record with bad signature
0x53	Outdated internal_record

6.6.7.6 Domain Controller transfer finish

```
void DC_transfer_confirm_message ( void );
```

This message shall be the final message to allow cleanup of change records.

Table 195: Domain Controller transfer finish message

Syntax	Bits	Identifier
DC_transfer_finish_message(){		
CPCM_protocol_message_type = 0x601A	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_transfer_finish_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x601A.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using SAC session key.

There are no protocol-specific error codes defined for this protocol call.

6.6.7.7 Become Domain Controller

```
void become_DC_message (
  ADSE_values()     ADSE_values
);
```

This message shall be sent by the Domain Controller to inform a Domain Member, a new Domain Controller (Transfer Case), or an additional Domain Controller (Split Case) of its new status.

Table 196 : Become Domain Controller message

Syntax	Bits	Identifier
become_DC_message(){		
CPCM_protocol_message_type = 0x6080	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
ADSE_values		ADSE_values()
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

Semantics for become_DC_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6080.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

ADSE_values: Carries the current state of the ADSE variables associated with an AD (see clause 5.4.19).

SAC_secret_signature: This is the message signature using SAC session key.

The possible CPCM protocol error codes specific to this protocol call are listed in table 197.

Table 197 : Become DC specific error codes

Error code	Meaning
0x89	Unexpected ADSE values

6.6.7.8 New Domain Controller

```
void new_DC_message ( void );
```

This message shall be sent by the CPCM Instance that is about to become a Domain Controller, following a DC Split or a DC Transfer.

Table 198 : New Domain Controller message

Syntax	Bits	Identifier
new_DC_message(){		
CPCM_protocol_message_type = 0x6081	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

Semantics for new_DC_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6081.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using SAC session key.

There are no protocol-specific error codes defined for this protocol call.

6.6.8 Domain Controller split

6.6.8.1 General

The Domain Controller split protocol is a transactional protocol applied in order to split the available ADM resources of the current Domain Controller to be shared with another CPCM Instance able to act as Domain Controller for the AD, as defined in TS 102 825-7 [i.9].

6.6.8.2 Domain Controller split begin

```
void DC_split_begin_message (
  ADID                          ADID,
  AD_capability()               capability,
  ADM_status                    status,
);
```

This message shall be sent to a Domain Controller to signal the start of a Domain Controller split operation.

Table 199: Domain Controller Split begin message

Syntax	Bits	Identifier
DC_split_begin_message(){		
CPCM_protocol_message_type = 0x601B	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
capability()	8	AD_capability()
status	64	ADM_status
[conditional_elements()]		
[optional_elements()]		
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_split_begin_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x601B.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

capability(): Carries the replication of the five AD and ADM capability fields of the CPCM Instance Certificate.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

The possible CPCM protocol error codes specific to this protocol call are listed in table 200.

Table 200: DC split begin specific error codes

Error code	Meaning
0x40	No Domain Controller is available
0x41	Requested Domain Controller is not available
0x42	Instance is not a Domain Controller
0x48	DC_remote_ceiling exceeded
0x49	DC_ceiling exceeded
0x4B	Rejection by ADMAAA
0x80	CPCM Instance is not AD aware
0x81	CPCM Instance is not a member of any AD
0x82	CPCM Instance is not a member of the indicated AD
0x83	CPCM Instance is not ADM capable
0x85	Instance is not Domain Controller capable
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x91	Certificate-message mismatch

6.6.8.3 Domain Controller Split ready

```
void DC_split_ready_message (
  ADID                                ADID,
  ADM_status                          status,
  ADSE_values()                       ADSE_values
);
```

This message shall be returned by a Domain Controller when prepared for a Domain Controller split operation.

Table 201: Domain Controller split ready message

Syntax	Bits	Identifier
DC_split_ready_message(){		
CPCM_protocol_message_type = 0x601C	16	bslbf
control_field = 0x0C	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
status	8	ADM_status
ADSE_values		ADSE_values()
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
AD_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_split_ready_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x601C.

control_field: This message shall be signed by both the SAC session key and the AD secret, hence this field shall be set to 0x0C.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

ADSE_values: Carries the current state of the ADSE variables associated with an AD (see clause 5.4.19).

SAC_secret_signature: This is the message signature using SAC session key.

AD_secret_signature: The signature of the AD Secret, used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 202.

Table 202: DC split ready specific error codes

Error code	Meaning
0x42	Instance is not a Domain Controller
0x80	CPCM Instance is not AD aware
0x83	CPCM Instance is not ADM capable
0x85	Instance is not Domain Controller capable
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x89	Unexpected ADSE values
0x91	Certificate-message mismatch

6.6.8.4 Domain Controller split commit

```
void DC_split_commit_message (
  ADSE_values()         ADSE_values
);
```

This message shall be sent to a Domain Controller for completion of a Domain Controller Split operation.

Table 203: Domain Controller split commit message

Syntax	Bits	Identifier
DC_split_commit_message(){		
CPCM_protocol_message_type = 0x601D	16	bslbf
control_field = 0x0C	8	bslbf
message_payload_length	16	uimsbf
ADSE_values		ADSE_values()
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
AD_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_split_commit_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x601D.

control_field: This message shall be signed by both the SAC session key and the AD secret, hence this field shall be set to 0x0C.

message_payload_length: The number of message_payload_bytes of the message.

ADSE_values: Carries the requested ADSE variables associated with the AD (see clause 5.4.19).

SAC_secret_signature: This is the message signature using the SAC session key.

AD_secret_signature: The signature of the AD Secret, used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 204.

Table 204: DC split commit specific error codes

Error code	Meaning
0x4D	Bad ADSE counts requested

6.6.8.5 Domain Controller split confirm

```
void DC_split_confirm_message (
  ADSE_values()             ADSE_values
  AD_internal_record        internal_record
);
```

This message shall provide the confirmation that Domain Controller shall now be split.

Table 205: Domain Controller split confirm message

Syntax	Bits	Identifier
DC_split_confirm_message(){		
CPCM_protocol_message_type = 0x601E	16	Bslbf
control_field = 0x08	8	Bslbf
message_payload_length	16	Uimsbf
ADSE_values		ADSE_values()
internal_record		AD_internal_record
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	Bslbf
}		

NOTE 1: Internal_record is the first Domain Internal Record of the new Domain Controller.

NOTE 2: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_split_confirm_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x601E.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

ADSE_values: Carries the current state of the ADSE variables associated with an AD (see clause 5.4.19).

internal_record: The internal record for the DC. See clause 5.4.22.

SAC_secret_signature: This is the message signature using SAC session key.

The possible CPCM protocol error codes specific to this protocol call are listed in table 206.

Table 206: DC split confirm specific error codes

Error code	Meaning
0x89	Unexpected ADSE values

6.6.8.6 Domain Controller split finish

```
void DC_split_finish_message ( void );
```

This message shall be the final message to allow cleanup of change records.

Table 207: Domain Controller split finish message

Syntax	Bits	Identifier
DC_split_finish_message(){		
CPCM_protocol_message_type = 0x601F	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_split_finish_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x601F.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using the SAC session key.

There are no protocol-specific error codes defined for this protocol call.

6.6.9 Domain Controller Merge

6.6.9.1 General

The Domain Controller merge protocol is a transactional protocol applied in order to merge the available ADM resources of another CPCM Instance with those of the initiating CPCM Instance acting as Domain Controller for the AD, as defined in TS 102 825-7 [i.9].

6.6.9.2 Domain Controller merge begin

```
void DC_merge_begin_message (
  ADID                  ADID,
  ADM_status            status
);
```

This message shall be sent to a Domain Controller to signal the start of a Domain Controller merge operation.

Table 208: Domain Controller Merge begin message

Syntax	Bits	Identifier
DC_merge_begin_message(){		
CPCM_protocol_message_type = 0x6020	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
status	8	ADM_status
[conditional_elements()]		
[optional_elements()]		
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_merge_begin_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6020.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

The possible CPCM protocol error codes specific to this protocol call are listed in table 209.

Table 209: DC merge begin specific error codes

Error code	Meaning
0x40	No Domain Controller is available
0x41	Requested Domain Controller is not available
0x42	Instance is not a Domain Controller
0x80	CPCM Instance is not AD aware
0x81	CPCM Instance is not a member of any AD
0x82	CPCM Instance is not a member of the indicated AD
0x83	CPCM Instance is not ADM capable
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x91	Certificate-message mismatch

6.6.9.3 Domain Controller merge ready

```
void DC_merge_ready_message (
  ADID                  ADID,
  ADM_status            status,
  ADSE_values()         ADSE_values
);
```

This message shall be sent by a Domain Controller when prepared for a Domain Controller merge operation.

Table 210: Domain Controller Merge ready message

Syntax	Bits	Identifier
DC_merge_ready_message(){		
CPCM_protocol_message_type = 0x6021	16	bslbf
control_field = 0x0C	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
status	8	ADM_status
ADSE_values		ADSE_values()
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
AD_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_merge_ready_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6021.

control_field: This message shall be signed by both the SAC session key and the AD secret, hence this field shall be set to 0x0C.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

ADSE_values: Carries the current state of the ADSE variables associated with an AD (see clause 5.4.19).

SAC_secret_signature: This is the message signature using SAC session key.

AD_secret_signature: The signature of the AD Secret, used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 211.

Table 211 : DC merge ready specific error codes

Error code	Meaning
0x42	Instance is not a Domain Controller
0x80	CPCM Instance is not AD aware
0x83	CPCM Instance is not ADM capable
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x89	Unexpected ADSE values
0x91	Certificate-message mismatch

6.6.9.4 Domain Controller Merge commit

```
void DC_merge_commit_message (
  ADID                              ADID,
  ADM_status                        status,
  ADSE_values()                     ADSE_values,
  AD_internal_record                internal_record
);
```

This message shall be sent to a Domain Controller to proceed to a Domain Controller merge operation.

Table 212: Domain Controller merge commit message

Syntax	Bits	Identifier
DC_merge_commit_message(){		
CPCM_protocol_message_type = 0x6022	16	bslbf
control_field = 0x0C	8	bslbf
message_payload_length	16	uimsbf
ADID	72	bslbf
status	8	ADM_status
ADSE_values		ADSE_values()
internal_record		AD_internal_record
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
AD_secret_signature	160	bslbf
}		

NOTE 1: internal_record embeds the current Domain Internal record of the Merging Domain Controller.

NOTE 2: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_merge_commit_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6022.

control_field: This message shall be signed by both the SAC session key and the AD secret, hence this field shall be set to 0x0C.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

ADSE_values: Carries the current state of the ADSE variables associated with an AD (see clause 5.4.19).

internal_record: The internal record for the DC see clause 5.4.22.

SAC_secret_signature: This is the message signature using SAC session key.

AD_secret_signature: The signature of the AD Secret, used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 213.

Table 213 : DC merge commit specific error codes

Error code	Meaning
0x42	Instance is not a Domain Controller
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x89	Unexpected ADSE values
0x91	Certificate-message mismatch

6.6.9.5 Domain Controller merge confirm

```
void DC_merge_confirm_message ( void );
```

This message shall provide the confirmation that Domain Controller merge has been completed.

Table 214: Domain Controller merge confirm message

Syntax	Bits	Identifier
DC_merge_confirm_message(){		
CPCM_protocol_message_type = 0x6023	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_merge_confirm_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6023.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using SAC session key.

There are no protocol-specific error codes defined for this protocol call.

6.6.9.6 Domain Controller merge finish

```
void DC_merge_finish_message ( void );
```

This message shall be the final message to allow cleanup of change records.

Table 215: Domain Controller merge finish message

Syntax	Bits	Identifier
DC_merge_finish_message(){		
CPCM_protocol_message_type = 0x6024	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_merge_finish_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6024.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using the SAC session key.

There are no protocol-specific error codes defined for this protocol call.

6.6.9.7 Merge Domain Controller

```
void DC_merge_DC_message ( void );
```

This message shall be sent by the Domain Controller to let another Domain Controller merge with it.

Table 216: Merge Domain Controller message

Syntax	Bits	Identifier
merge_DC_message(){		
CPCM_protocol_message_type = 0x6082	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

Semantics for merge_DC_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6082.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using SAC session key.

There are no protocol-specific error codes defined for this protocol call.

6.6.9.8 Domain Controller Merged

```
void DC_merged_message (
  ADSE_values()     ADSE_values
);
```

This message shall be sent by a CPCM Instance merging its Domain Controller function with another one.

Table 217: Domain Controller merged message

Syntax	Bits	Identifier
DC_merged_message(){		
CPCM_cprotocol_message_type = 0x6083	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
ADSE_values		ADSE_values()
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

Semantics for DC_merged_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6083.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

ADSE_values: Carries the current state of the ADSE variables associated with an AD (see clause 5.4.19).

SAC_secret_signature: This is the message signature using SAC session key.

The possible CPCM protocol error codes specific to this protocol call are listed in table 218.

Table 218: DC merged specific error codes

Error code	Meaning
0x89	Unexpected ADSE values

6.6.10 Domain Controller rebalance

6.6.10.1 General

The Domain Controller rebalance protocol is a transactional protocol applied in order to rebalance the ADM resources of an AD, as defined in TS 102 825-7 [i.9].

6.6.10.2 Domain Controller rebalance begin

```
void DC_rebalance_begin_message (
  ADID                  ADID,
  ADM_status            status
);
```

This message shall be sent to a Domain Controller to signal the start of a Domain Controller Rebalance operation.

Table 219: Domain Controller rebalance begin message

Syntax	Bits	Identifier
DC_rebalance_begin_message(){		
CPCM_protocol_message_type = 0x6025	16	bslbf
control_field = 0x00	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
status	8	ADM_status
[conditional_elements()]		
[optional_elements()]		
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_rebalance_begin_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6025.

control_field: This message shall be neither signed nor encrypted, hence this shall be set to 0x00.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

The possible CPCM protocol error codes specific to this protocol call are listed in table 220.

Table 220: DC rebalance begin specific error codes

Error code	Meaning
0x40	No Domain Controller is available
0x41	Requested Domain Controller is not available
0x42	Instance is not a Domain Controller
0x80	CPCM Instance is not AD aware
0x81	CPCM Instance is not a member of any AD
0x82	CPCM Instance is not a member of the indicated AD
0x83	CPCM Instance is not ADM capable
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x91	Certificate-message mismatch

6.6.10.3 Domain Controller rebalance ready

```
void DC_rebalance_ready_message (
  ADID                  ADID,
  ADM_status            status,
  ADSE_values()         ADSE_values
);
```

This message shall be returned by a Domain Controller when prepared for a Domain Controller Rebalance operation.

Table 221: Domain Controller Rebalance ready message

Syntax	Bits	Identifier
DC_rebalance_ready_message(){		
CPCM_protocol_message_type = 0x6026	16	bslbf
control_field = 0x0C	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
status	8	ADM_status
ADSE_values		ADSE_values()
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
AD_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_rebalance_ready_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6026.

control_field: This message shall be signed by both the SAC session key and the AD secret, hence this field shall be set to 0x0C.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

ADSE_values: Carries the current state of the ADSE variables associated with an AD (see clause 5.4.19).

SAC_secret_signature: This is the message signature using SAC session key

AD_secret_signature: The signature of the AD Secret, used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 222.

Table 222: DC rebalance ready specific error codes

Error code	Meaning
0x42	Instance is not a Domain Controller
0x80	CPCM Instance is not AD aware
0x83	CPCM Instance is not ADM capable
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x89	Unexpected ADSE values
0x91	Certificate-message mismatch

6.6.10.4 Domain Controller rebalance commit

```
void DC_rebalance_commit_message (
  ADID                  ADID,
  ADM_status            status,
  ADSE_values()         ADSE_values
);
```

This message shall be sent to a Domain Controller so signal completion of a Domain Controller rebalance operation.

Table 223: Domain Controller rebalance commit message

Syntax	Bits	Identifier
DC_rebalance_commit_message(){		
CPCM_protocol_message_type = 0x6027	16	bslbf
control_field = 0x0C	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
status	8	ADM_status
ADSE_values		ADSE_values()
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
AD_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_rebalance_commit_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6027.

control_field: This message shall be signed by both the SAC session key and the AD secret, hence this field shall be set to 0x0C.

message_payload_length: The number of message_payload_bytes of the message.

ADID: The AD Identifier of the AD.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

ADSE_values: Carries the requested ADSE variables associated with the AD (see clause 5.4.19).

SAC_secret_signature: This is the message signature using SAC session key.

AD_secret_signature: The signature of the AD Secret, used to securely verify the AD membership through knowledge of the AD Secret.

The possible CPCM protocol error codes specific to this protocol call are listed in table 224.

Table 224: DC rebalance commit specific error codes

Error code	Meaning
0x42	Instance is not a Domain Controller
0x4D	Bad ADSE counts requested
0x87	ADID mismatch
0x88	The indicated AD has been revoked
0x89	Unexpected ADSE values
0x91	Certificate-message mismatch

6.6.10.5 Domain Controller rebalance confirm

```
void DC_rebalance_confirm_message ( void );
```

This message shall provide the confirmation that Domain Controller rebalance has been completed.

Table 225: Domain Controller Rebalance confirm message

Syntax	Bits	Identifier
DC_rebalance_confirm_message(){		
CPCM_protocol_message_type = 0x6028	16	bslbf
control_field = 0x0C	8	bslbf
message_payload_length	16	uimsbf
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_rebalance_confirm_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6028.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using the SAC session key.

There are no protocol-specific error codes defined for this protocol call.

6.6.10.6 Domain Controller rebalance finish

```
void DC_rebalance_finish_message ( void );
```

This message shall be the final message to allow cleanup of change records.

Table 226: Domain Controller rebalance finish message

Syntax	Bits	Identifier
DC_rebalance_finish_message(){		
CPCM_protocol_message_type = 0x6029	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
[conditional_elements()]		
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

NOTE: The delegating_instance_id element may be present within the conditional_elements.

Semantics for DC_rebalance_finish_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6029.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using SAC session key.

There are no protocol-specific error codes defined for this protocol call.

6.6.10.7 Rebalance Domain Controller

```
void rebalance_DC_message (
  ADSE_values()     ADSE_values
);
```

This message shall be sent by the Domain Controller to Rebalance its ADSE values with another one.

Table 227: Rebalance Domain Controller message

Syntax	Bits	Identifier
rebalance_DC_message(){		
CPCM_protocol_message_type = 0x6084	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
ADSE_values		ADSE_values()
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

Semantics for rebalance_DC_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6084.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

ADSE_values: Carries the current state of the ADSE variables associated with an AD (see clause 5.4.19).

SAC_secret_signature: This is the message signature using SAC session key.

The possible CPCM protocol error codes specific to this protocol call are listed in table 228.

Table 228 : Rebalance DC specific error codes

Error code	Meaning
0x89	Unexpected ADSE values

6.6.10.8 Domain Controller rebalanced

```
void DC_rebalanced_message ( void );
```

This message shall be sent by a Domain Controller to confirm it has just Rebalanced its ADSE values.

Table 229: Domain Controller rebalanced message

Syntax	Bits	Identifier
DC_rebalanced_message(){		
CPCM_protocol_message_type = 0x6085	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

Semantics for rebalance_DC_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x6085.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

SAC_secret_signature: This is the message signature using SAC session key.

There are no protocol-specific error codes defined for this protocol call.

6.6.11 ADMAAA management

6.6.11.1 General

The ADMAAA management protocol enables the Domain Controller to verify with the AAA whether an ADM operation is allowed, as defined in TS 102 825-7 [i.9].

6.6.11.2 ADMAAA tool request

```
void ADMAAA_tool_request_message (
  ADID                ADID
  ADM_status          status,
  AD_operation        protocol,
  AD_condition        condition,
  ADSE_values()       ADSE_values
);
```

This message is sent by the Domain Controller to test if a Join or Split is allowed by that AAA.

Table 230: ADM AAA Tool request message

Syntax	Bits	Identifier
ADMAAA_tool_request_message(){		
CPCM_protocol_message_type = 0x60F0	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
ADID	72	ADID
status	8	ADM_status
protocol	8	AD_operation
condition	8	AD_condition
ADSE_values	72	ADSE_values()
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

Semantics for ADMAAA_tool_request_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x60F0.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

status: Indicates if the instance is Blank, an AD member, Local Master, Domain Controller or both Local Master and Domain Controller. See clause 6.6.2.2.

operation: See clause 6.6.2.4.

condition: See clause 6.6.2.3.

ADSE_values: Carries the current state of the ADSE variables associated with an AD (see clause 5.4.19).

SAC_secret_signature: This is the message signature using SAC session key.

The possible CPCM protocol error codes specific to this protocol call are listed in table 231.

Table 231: ADMAAA tool request specific error codes

Error code	Meaning
0x42	Instance is not a Domain Controller
0x4B	Rejection by ADMAAA
0x80	CPCM Instance is not AD aware
0x81	CPCM Instance is not a member of any AD
0x82	CPCM Instance is not a member of the indicated AD
0x83	CPCM Instance is not ADM capable
0x88	The indicated AD has been revoked
0x91	Certificate-message mismatch

6.6.11.3 ADMAAA tool response

```
void ADMAAA_tool_response_message (
  ADSE_values()     ADSE_values
);
```

This message shall be in response to an ADMAAA tool request.

Table 232: ADMAAA tool response message

Syntax	Bits	Identifier
ADMAAA_tool_response_message(){		
CPCM_protocol_message_type = 0x60F1	16	bslbf
control_field = 0x08	8	bslbf
message_payload_length	16	uimsbf
ADSE_values	8	ADSE_values()
[optional_elements()]		
SAC_secret_signature	160	bslbf
}		

NOTE: ADMAAA may authorize the operation without authorizing any ceiling increase. In that case, only local counts are updated.

Semantics for ADMAAA_tool_response_message:

CPCM_protocol_message_type: According to tables 40 and 142 this shall be set to 0x60F1.

control_field: This message shall be signed by the SAC session key, hence this field shall be set to 0x08.

message_payload_length: The number of message_payload_bytes of the message.

ADSE_values: Carries the current state of the ADSE variables associated with an AD (see clause 5.4.19).

SAC_secret_signature: This is the message signature using SAC session key.

The possible CPCM protocol error codes specific to this protocol call are listed in table 233.

Table 233 : ADMAAA tool response specific error codes

Error code	Meaning
0x89	Unexpected ADSE values

6.7 CPCM Extension messages

```
void CPCM_extension_message (
  identifier            CPCM_extension_identifier,
  extension_data        CPCM_extension_data
);
```

Facilities for the management of CPCM System extensions are not included in this version of the specification, but a generic CPCM protocol message type for communication with CPCM extensions is defined for future use. The protocol is depicted in Figure 35 and Table 234 shows the syntax of this generic message.

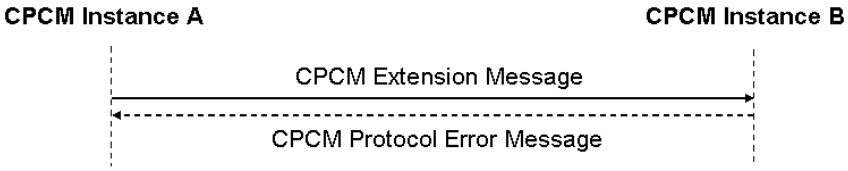

Figure 35: CPCM extension message protocol

Table 234: CPCM extension message

Syntax	Bits	Identifier
CPCM_extension_message(){		
CPCM_protocol_message_type = 0xFFFF	16	bslbf
control_field	8	bslbf
message_payload_length	16	uimsbf
CPCM_extension_identifier	8	bslbf
CPCM_extension_data		bslbf
[SAC_secret_signature || AD_secret_signature]	160	bslbf
}		

NOTE: The message may be encrypted with the AD Secret, the SAC Secret, or may be in the clear depending on the control field value.

Semantics for CPCM_extension_message:

CPCM_protocol_message_type: This shall be set to 0xFFFF.

control_field: This field shall be set regarding the protection level required by the CPCM extension data.

message_payload_length: The number of message_payload_bytes of the message.

CPCM_extension_identifier: This field gives the identifier of the CPCM extension.

CPCM_extension_data: CPCM extension data.

The possible CPCM protocol error codes specific to this protocol call are listed in table 235.

Table 235: CPCM extension message specific error codes

Error code	Meaning
0x20	Unknown extension
0x21 to 0x3F	Reserved for future use
0x40 to 0xFF	Extension-specific error codes

6.8 Private message

```
void private_data_message (
  identifier               private_data_identifier,
  data                     private_data
);
```

The private date message protocol provides a generic method for the communication of private data between CPCM Instances using the CPCM protocol framework. The protocol is depicted in figure 36 and table 236, shows the syntax of this generic message.

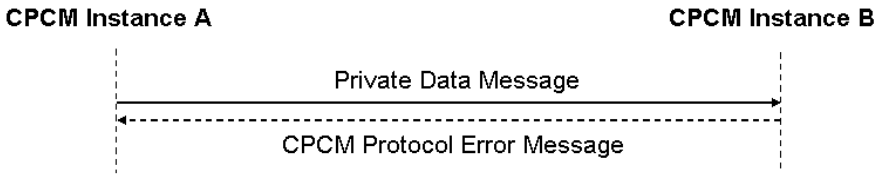

Figure 36: CPCM private data message protocol

Table 236: Private data message

Syntax	Bits	Identifier
private_data_message(){		
CPCM_protocol_message_type = 0xFFFE	16	bslbf
control_field	8	bslbf
message_payload_length	16	uimsbf
private_data_identifier	16	uimsbf
private_data		bslbf
[SAC_secret_signature || AD_secret_signature]	160	bslbf
}		

NOTE: The message may be encrypted with the AD Secret, the SAC Secret, or may be in the clear depending on the control field value.

Semantics for CPCM_extension_message:

CPCM_protocol_message_type: This shall be set to 0xFFFE.

control_field: This field shall be set regarding the protection level required by the private data.

message_payload_length: The number of message_payload_bytes of the message.

private_data_identifier: This field gives the identifier of the private data.

private_data: This is the private data payload.

The possible CPCM protocol error codes specific to this protocol call are listed in table 237.

Table 237: CPCM private message specific error codes

Error code	Meaning
0x20	Unknown private data identifier
0x21 to 0x3F	Reserved for future use
0x40 to 0xFF	Private error codes

7 CPCM Content Management

7.1 General

This clause lays out the generic framework of the management of CPCM Content within the CPCM System.

The CPCM Reference Model (TS 102 825-2 [i.5]) sets a basic framework for CPCM Content management in terms of each of the five CPCM abstract Functional Entities - Acquisition, Storage, Processing, Consumption and Export - individually, but here it is specified in terms of how CPCM Content is managed and handled within the CPCM System, taking into account the following aspects:

- The function of CPCM within a home network ecosystem and over application-specific physical interfaces;
- The application of the CPCM Security Toolbox for CPCM Content transactions;

- Methods to handle CPCM Content Licences in relation to the CPCM Content to which they apply; and
- The link between device capabilities (securely advertised in the CPCM Device Certificate) and the Authorized Usage of the CPCM Content that the CPCM Device handles.

The generic part of the CPCM System specification is agnostic as regards the actual format of CPCM Content, so this clause deals only generically with CPCM Content Management. Aspects pertaining to the working of the CPCM System with particular content formats are defined in TS 102 825-9 [i.3].

Content storage is a key feature of any content handling and management scheme, but this clause deals only with the generic concept of CPCM Content Storage. Aspects pertaining to the working of the CPCM System with particular content storage formats are defined in TS 102 825-9 [i.3].

TR 102 825-11 [i.7] contains informative descriptions of various content usage scenarios in the CPCM System, utilizing the individual CPCM Content management aspects documented in this clause. The set of scenarios is not exhaustive but it aims to provide a broad range of realistic content provision and management scenarios that CPCM is intended to enable.

The set of CPCM Usage Rules and their corresponding coding in USI provide a common basis for many content provision and usage scenarios within the CPCM System. They also enable content exchange to and from other content protection systems via Usage Rules mappings.

Naturally CPCM is closely linked with home networking, but does not prescribe any particular requirements from its protocols or physical interfaces. CPCM is intended to work in a way that is transparent to the "home network", by being implemented as a secure resource controlled by the device's application on each compliant device, and by using the basic resources of the home network to carry out the necessary CPCM Communications between CPCM Devices as appropriate for those Devices.

7.2 CPCM Device and Content discovery

It is intended that CPCM works as transparently as possible for the user. In a typical scenario CPCM will function within a home network ecosystem that can host both CPCM Devices and Content and other non-CPCM devices and content. CPCM Devices will use the usual discovery mechanisms provided by the home network ecosystem in order to make the CPCM Content under their control available to the user. This is facilitated by the addition of CPCM-specific device attributes within the home network discovery process. The following properties, stored securely in the CPCM Instance Certificate (CIC), shall be replicated in the home network device discovery process, either as an extension to the normal device attributes, or in a separate CPCM-specific discovery process:

- CPCM Version.
- CPCM Instance Certificate Identifier.
- C&R regime Mask.
- LSA Capable.
- Content Handling Capability.
- Absolute Time Capable.
- Geographical Aware.
- AD Aware.
- ADM Capable.
- ADSE Countable.
- ADM DC Capable.
- ADM LM Capable.

CPCM Content could be made available to the user as part of their normal home network navigation interfaces, whereby CPCM Content could be listed along with any other content that the networked device is able to discover.

Alternatively the device could list CPCM Content in a separate menu. These aspects are not specified by the CPCM System but left open to implementation.

Any CPCM Device that maintains control over any CPCM Content items will naturally be able to advertise the availability of that content to the user of that device via its user interface (if it has one).

In the case of networked CPCM Devices, a different CPCM Device connected within the home network ecosystem will be able to discover the presence of CPCM Content maintained by enquiring the content directory service (or similar) facility of the CPCM Device hosting that CPCM Content.

CPCM Content may exist as well on non-CPCM devices. This is for instance the case in bit bucket scenarios. The device hosting is responsible for advertising the Content using applicable Content Discovery protocols (e.g. DVB-HN).

Self-contained CPCM Content - any number of CPCM Instances can be aware of such content and enable it to be discovered by other CPCM Devices in the home network / within the CPCM System.

Non-CPCM devices may also be able to discover CPCM Content and to perform some basic operations on it (e.g. delete it, request a copy or a Move). These operations will be performed by using appropriate content management protocols in the home network ecosystem in operation, and they will be relayed to CPCM that will handle them if allowed by the CPCM Content's Authorized Usage.

Devices should advertise CPCM-relevant properties of any CPCM Content that they make available to the user. That is, relevant attributes of CPCM Content should be presented to the user in their network content guide in a user-friendly manner, for example, utilizing icons to signify particular Authorized Usage attributes.

Thus the following metadata of each CPCM Content item should be made available in the network content directory service (or similar) so that other CPCM Instances advertising that content will be able to inform the user about its CPCM-relevant properties:

- USI (Usage State Information) and Auxiliary Data - to allow possible indication to the user of the Authorized Usage of that Content item;
- AD Identifier, if applicable - to indicate to which AD the Content item is bound.

Based on the above information, the content guide may display all available content (regardless whether the user may consume it) or filter out Content that is not accessible to the user (e.g. Content from different AD that is not MLocal or VLocal, expired content, etc.).

The metadata stored in the network content guide is naturally not secure, but is provided by CPCM for information. The Authorized Usage of any Content item shall be securely applied on the basis of the USI stored in the Content Licence in combination with the Auxiliary Data.

Figure 37 shows the generic model for CPCM Content discovery in a home network ecosystem. Nominal networked devices A and B are both CPCM Devices, and are thus capable of discovering CPCM Content that the other makes available to the home network content discovery mechanism.

Device C is a non-CPCM device, but one that is holding CPCM Content items. There are three possibilities for the management of the Content Licences associated with such Content items:

- The Content Licence may be retained by the Device that Stored the Content item on it (Device B in figure 37). In that case, no Content Licence is present with Content item. Device B may choose to associate its CPCM Certificate Instance Identifier to let other CPCM Instances know that Content item is under its control and only accessible by its intermediary. This is achieved by the inclusion of the optional last_CL_issuer field in the Content Licence (see clause 5.4.4).
- The Content Licence may be recorded together with the Content item but still under the protection of the Device Secret of the Device that Stored the Content item on it. As in the above case, the Content item is accessible only via Device B.
- The Content Licence may be under the protection of the AD Secret (bit bucket scenario). In that case, the Content item is directly accessible by any CPCM Instance that is a member of the same AD. CPCM Instances that are not members of the AD may access the Content item (if allowed by the USI) only through a CPCM Instance that is a member of this AD.

CPCM secure communications interface CPCM-1 is not depicted in this diagram because it does not need to have come into play for the purpose of CPCM Content discovery.

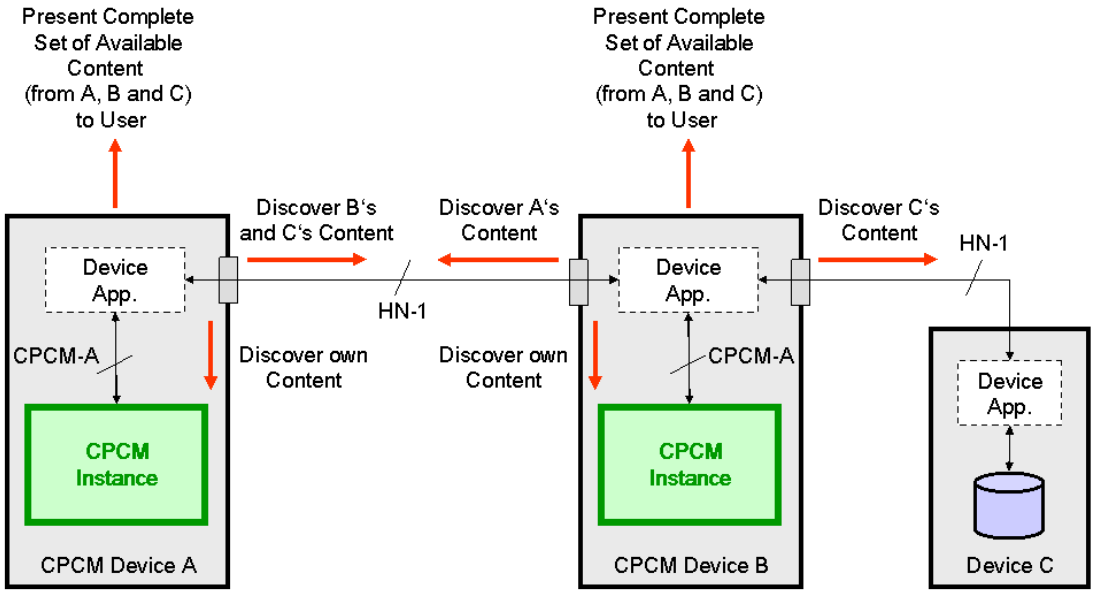

Figure 37: Generic model for CPCM Content discovery

7.3 Inter-Device CPCM Content exchange

Once CPCM Content items have been discovered via the home network, the user can usually select a Content item from his device UI, for example to view, or Consume it. Then the Device Application, as depicted generically in figure 37, triggers its CPCM Instance to take care of CPCM-specific management of the CPCM Content transfer.

Basically, the home network application of the CPCM Device handles content transport requests, and the CPCM Instance handles the corresponding CPCM Content Licence handling. These two sets of functionalities are essentially independent of each other. It is up to the CPCM Device implementation to ensure the correct and consistent mapping of the home network ecosystem with the CPCM System.

CPCM does not rely on the home network ecosystem's awareness of content-operations - play, record, process, etc., rather, CPCM Content operations are communicated independently of any equivalent home network system signalling, in the CPCM Content Licence Exchange protocols. Thus CPCM in effect decouples itself somewhat from the home network ecosystem. This benefits the security of CPCM Content operations, while imposing a common scheme for CPCM Content Management, to which various home network ecosystems shall adapt.

Figure 38 shows an example of a generic operational sequence for a CPCM Content item transfer between two CPCM Instances, also separating the home network interactions from those of the CPCM System, which are of course also communicated over the home network, but only the CPCM Instance parts of each device will be privy to that. In this example, the Content Licence is exchanged via a SAC and is not embedded in the Content item.

In this diagram all non-CPCM interactions are italicized. The generic sequence diagram contains four intra-CPCM process steps which form the basis of CPCM System operation:

1) SAC establishment (step 1 in figure 38) - specified in the corresponding CPCM protocols clause 6.4.2.

2) CPCM Content Licence exchange (step 2 in figure 38)- specified in the corresponding CPCM protocols clause 6.5.3.

3) CPCM Content Licence Verification (step 3 in figure 38) - specified below in clause 8.4. This includes the CPCM revocation check and verification of the Authorized Usage, specified below in clause 7.5.

4) CPCM ancillary information exchange (step 4 in figure 38) - the verification of the Content item's Authorized Usage, defined by the USI within the Content Licence, may necessitate the exchange of CPCM ancillary information e.g. secure time or geographic location.

There are many variations of this generic sequence diagram, depending on the content operation requested, the Authorized Usage settings of the Content item, or the current status of each of the CPCM Instances.

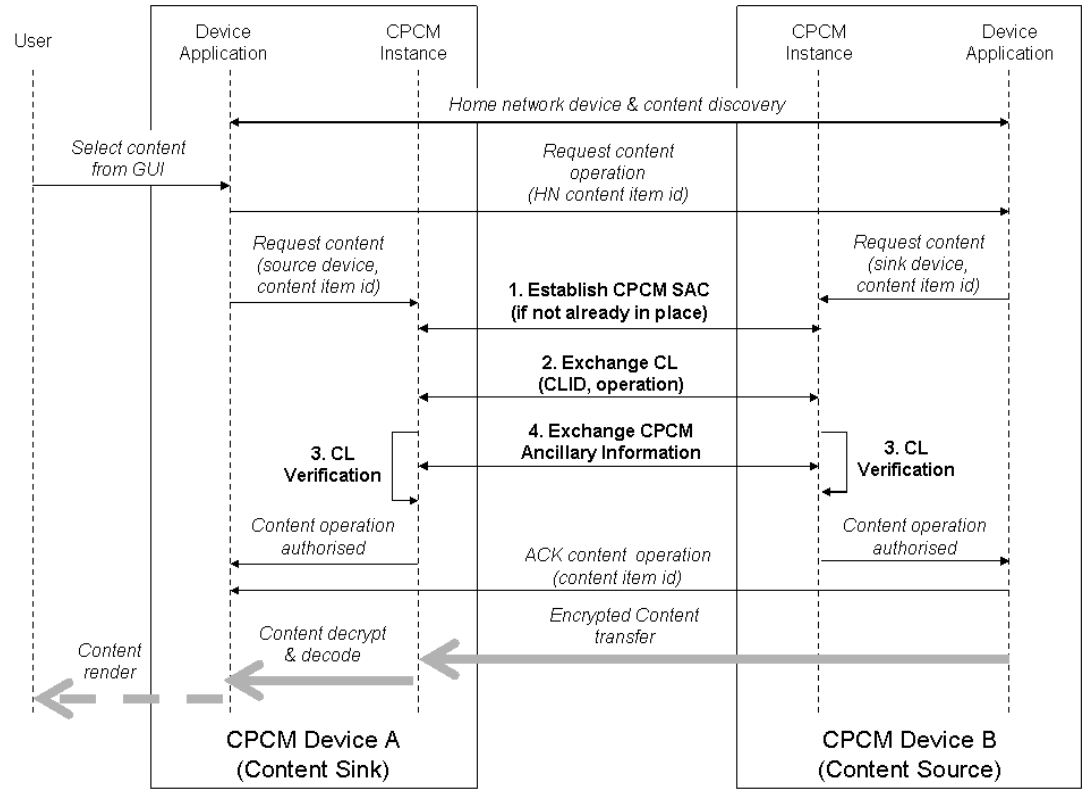

Figure 38: Generic model for CPCM Content exchange

The following clauses deal with CPCM Content management issues specifically relevant to each of the five CPCM Functional Entities.

7.4 CPCM functional entity behaviour

7.4.1 Acquisition

When Input Content is Acquired by an Acquisition Point, the following shall be generated:

- A CPCM Content Licence, as described in clause 8.2.

NOTE: 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.

- CPCM Auxiliary Data, if necessary and as defined in TS 102 825-3 [i.11], including the original usage rules signalling and other data as applicable.

- CPCM Content: unless CPCM Content is not marked DNCS, CPCM content is scrambled as described in clause 7.6, by applying the chosen descrambling_key and CPCM_descrambler_information as stored in the Content Licence.

Before sending CPCM Content to another CPCM Instance, the Acquisition Point shall first enforce the USI as described in clause 7.5.

When content is delivered as Protected Input Content (e.g. from a CA or DRM system), the end of a service event shall be signalled to the Acquisition Point in the following cases:

- A general security requirement stipulates that not too much content gets scrambled with the same key, therefore the end of a service event shall always cause a change of descrambling key. The C&R regime may also impose a maximum duration for the validity of the descrambling key. If this maximum period expires during a service event that is being Sourced as a CPCM Content item then a change of the descrambling key shall be forced by applying the procedure specified in clause 7.6.2.
- To terminate the applicability of SVCA controls.
- To signal that Remote Access can be enabled based upon one control parameter of the RAR.

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

7.4.2 Processing

When receiving CPCM Content, a Processing Entity shall do the following:

- Verify the Content Licence, as described in clause 8.4.
- Enforce the USI, as described in clause 7.5.
- Check whether it will be authorized to send the Processed CPCM Content to the Destination CPCM Instance (see clause 7.5.2).
- Process the CPCM Content. This processing may imply CPCM Content descrambling. In such a case, once processing has been achieved, it re-scrambles the content. The C&R regime may require in some cases to change the descrambling key. In such case, the new key shall be randomly chosen and a new Content Licence shall be issued (see clause 8.2). A Processing Entity may only change the descrambler_information and descrambling_key fields in the Content Licence.
- Send the processed CPCM Content to the Destination CPCM Instance.

A CPCM Instance that Processes a CPCM Content item shall not advertise the availability of the Content item that it is Processing to other CPCM Instances. Only the Processed Content item may be advertised to other CPCM Instances.

A CPCM Instance that Processes a CPCM Content item shall register the Sink or Source CPCM CIC identifier passed in the corresponding Content Licence exchange protocol call and apply the same Sink or Source CPCM CIC identifier in any subsequent exchange of the CL for the same Content Item to another CPCM Instance.

7.4.3 Export

When receiving CPCM Content, an Export Point shall do the following:

- Verify the Content Licence, as described in clause 8.4.
- Enforce the USI, as described in clause 7.5. If image_constraint is asserted and the content is high resolution, the Export Point (via a Processing Entity) shall constrain the resolution to standard definition.
- If Export (towards a Trusted CPS, a Controlled CPS or beyond CPCM Trust) is authorized and if applicable (i.e. if Content is not marked DNCS), descramble the Content.
- Proceed with the Content Export.

Exported Content (C.3) can be transported over the home network or over a CPCM application-specific interface.

A CPCM Instance that Exports a CPCM Content item shall not advertise the availability of that Content item to other CPCM Instances.

A CPCM Instance that Exports a CPCM Content item shall discard the obtained Content Licence for that Content item once Export has been completed. A renewed Export or a Consumption operation on the same Content item shall occur only after the associated Content Licence has been re-obtained from the Source CPCM Instance.

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

7.4.4 Consumption

When receiving CPCM content, a Consumption Point shall do the following:

- Verify the Content Licence, as described in clause 8.4.
- Enforce the USI, as described in clause 7.5.
- If Consumption is authorized, descramble the Content if applicable (i.e. if the Content item is not marked DNCS).
- Proceed to Consume the Content.

A CPCM Instance that Consumes a CPCM Content item shall not advertise the availability of that Content item to other CPCM Instances.

A CPCM Instance that Consumes a CPCM Content item shall 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 shall occur 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 shall map certain fields of the Content Licence and/or CPCM Auxiliary Data to that other CPS as required and defined by the C&R regime.

7.4.5 Storage

When receiving CPCM content, a Storage Entity shall do the following:

- verify the CL, as described in clause 8.4.
- enforce the USI, as described in clause 7.5 and, if required, change the USI to indicate that a new Copy has been generated (see clause 8.2.2).
- If applicable (e.g. if the CL was received through a SAC), protect the CL (see clause 8.6.1).
- Store the CPCM Content item.

When delivering CPCM Retrieved Content, a Storage Entity shall do the following:

- verify the CL as described in clause 8.4.
- enforce the USI as described in clause 7.5. As needed, it issues a new CL. A Storage Entity may change only certain elements of the USI in the CL (see clause 8.2.2.).
- If applicable (e.g. if the CL was protected using Device key), protect the CL (see clause 8.6.1).
- deliver the CPCM Retrieved Content.

A CPCM Instance that Stores a CPCM Content item shall advertise the availability of that Content item to other CPCM Instances if that Content item is intended to be accessible by other CPCM Instances.

7.5 USI enforcement

7.5.1 General

CPCM Content shall only be Acquired, Processed, Stored, Exported, and Consumed or handled in any other manner in accordance with the Authorized Usage, indicated by its USI, possibly in conjunction with its CPCM Auxiliary Data.

Most controls are performed by Sink CPCM Instances before they handle the CPCM Content. However, some controls must be performed by the Source CPCM Instance before the CPCM Content item is sent to another CPCM Instance.

7.5.2 Controls prior to Content transfer

A CPCM Instance shall not send, transfer or otherwise propagate Content unless all of the following controls of this clause and its subclauses allow such action for such Content. For avoidance of doubt, unless explicitly stated differently, any disallowance of an action below per one USI field overrides all other permissions for such action that might be allowed per other USI fields.

7.5.2.1 Copy and Movement control

7.5.2.1.1 Copy Control Not Asserted

If CPCM Content is marked Copy Control Not Asserted, content may be sent to a CPCM Instance or any other non-CPCM device.

7.5.2.1.2 Copy Once

Copy Once Content may only be sent by an Acquisition Point or a Processing Entity. A Storage Entity may never send Copy Once Content. Copy Once Content may only be sent to a CPCM Instance and never to a non-CPCM device.

If a CPCM Acquisition Point or a Processing Entity Sources more than one instance of Copy Once Content, then at most one of those instances may be marked Copy Once and the others must be marked Copy No More.

NOTE: If Content is aimed at being Consumed or Exported, Content should preferably be sent as Copy No More to enable a later recording of a different stream of the content that might be Sourced by the AP or PE.

7.5.2.1.3 Copy No More

CPCM Content that is marked "Copy No More" may be Sourced (with relevant CL protection) to another CPCM Instance but never to a Storage Entity, and never to a non-CPCM device.

CPCM Content that is marked "Copy No More" may only be Moved from one Storage Entity to another Storage Entity by using the Move Protocol (see clause 7.7).

7.5.2.1.4 Copy Never and Zero Retention

CPCM Content that is marked "Copy Never" may only be Sourced to another CPCM Instance (with relevant CL protection) but never to a Storage Entity, and never to a non-CPCM device.

Only Acquisition Points or Processing Entities may Source "Copy Never" content.

CPCM Content that is marked "Copy Never with Zero Retention not asserted" may only be Moved from one Storage Entity to another Storage Entity by using the Move Protocol (see clause 7.7).

7.5.2.2 Consumption Control

7.5.2.2.1 Viewable

If "Viewable" is asserted, no specific control is required before Sourcing the Content. Content may be sent to CPCM Instances and non-CPCM devices.

If "Viewable" is not asserted, content may only be Sourced to Processing Entities, Storage Entities and non-CPCM devices.

7.5.2.2.2 View Window Activated

This USI does not require any specific control before Sourcing the Content. Content may be Sourced to CPCM Instances and non-CPCM devices.

7.5.2.2.3 View Period Activated

"View Period Activated" (VPA) Content may only be sent by an Acquisition Point or a Processing Entity. A Storage Entity may not Source View Period Activated Content, unless within a Content Move protocol. "View Period Activated" Content may only be sent to a CPCM Instance.

Before sending content marked "View Period Activated", an Acquisition Point or Processing Entity first analyses what is the intent of the receiving CPCM Instance. If at least one CPCM Instance requests the Content for Consuming or Exporting it or if more than one CPCM Instance requests the content, the View Period starts and the sending CPCM Instance shall first re-issue a new CL (see below). Otherwise, the Acquisition Point or the Processing Entity just sends the content to the unique Storage Entity or Processing Entity that requests it.

A Storage Entity may Move a VPA content using Content Move protocol. Upon copy, consumption or export request, it shall first re-issue a new CL.

In order to re-issue a CL, a CPCM Instance first computes the View Window Start (current time) and the View Window End (current time augmented by View Window Period). The CPCM Instance shall to that end be secure absolute time capable or have access to secure absolute time using protocol described in TS 102 825-5 [i.8]. If the CPCM Instance does not have access to secure time, it shall not send the content for consuming or exporting purposes and shall send or Move the content to at most one Storage Entity or Processing Entity. Otherwise, the CPCM Instance clears VPA and VP and does one of the following:

- If Content was not marked "View Window Activated" (VWA), it asserts VWA with View Window being the one computed earlier.
- If Content was marked VWA, it computes the new View Window as:
  - View Window Start is the maximum of previous View Window Start and the current time.
  - View Window End is the minimum of previous View Window End and the current time augmented with the View Period.

NOTE: If View Window had not yet started, implementations may also choose to not handle the Content.

The CPCM Instance then sends the content with the newly issued Licence.

7.5.2.2.4 Simultaneous View Count Activated

An Acquisition Point delivering Content marked SVCA may deliver Content to any number of Storage Entities or Processing Entities or non-CPCM Devices. In such case, corresponding Content Licence shall be delivered using the *CPCM_put_CL_message*, specified in clause 6.5.3.5.

When the original Content transmission is over, the Acquisition Point issues a new Content Licence for which SVCA is not asserted and delivers it to each Storage Entity to which the Content was originally delivered. (NB. The Content may have been delivered via a Processing Entity).

Upon receipt of the *CPCM_get_permission_message* from a Storage Entity or a Processing Entity, the Acquisition Point refuses the permission using the message *CPCM_get_permission_response_message*.

An Acquisition Point may deliver up to `simultaneous_view_count` Content Licences to Consumption Points or Export Points using the *CPCM_put_CL_message*, under the SAC protection.

Upon receipt of the *CPCM_get_permission_message* or upon CL request from a Consumption Point or Export Point, the Acquisition Point shall do the following:

- If the Content is finished, it issues a new Content Licence where SVCA is not asserted and delivers the new CL to the requesting CPCM Instance and to each Storage Entity (possibly through a Processing Entity) to which the CL was originally sent, using message *CPCM_put_CL_message*.
- Else, it checks whether the total number of granted permissions and CL delivered to Consumption Points and Export Points under SAC protection is less than `simultaneous_view_count`. If so, it grants the permission through the *CPCM_get_permission_response_message*. or it delivers the requested CL using the *CPCM_get_CL_response_message*.

- Else, it sends to each Consumption Point and Export Point to which a permission or a CL under SAC protection was previously delivered a *CPCM_get_content_item_status_request_message* carrying the CLID. If a positive answer is received from each of them, it sends a message denying the permission, using the message *CPCM_get_permission_response_message* or returns an error to the CL request.

- Else, permission may be granted. To that end, the Acquisition Point picks a new randomly selected descrambling key, issues a new Content Licence and sends the new CL to each of the requesting CPCM Instances and to each of the CPCM Instances hosting the Consumption Points and Export Points to which a permission or a CL under SAC protection was previously delivered using the *CPCM_put_CL_message* and that responded positively to the *CPCM_get_content_item_status_request_message*. This new Content Licence shall also be sent to each Storage Entity (which may have received the Content via a Processing Entity) using the *CPCM_put_CL_message*.

Storage Entities and Processing Entities may send Content marked SVCA to any number of other CPCM Instances or non-CPCM devices. If they received several CL for the content, they shall re-transmit all the received Licences as well.

The Acquisition Point shall maintain a link between the Content and one or more CLID in order to determine the number of current Content Consumptions or Exports. How this link is maintained is beyond the scope of the present document.

7.5.2.3 Propagation Control

The *CPCM_get_CL_message* (if any) indicates what the destination Sink CPCM Instance intends to do with the Content (i.e. Processing, Export, Consumption or Storage).

The Source assesses whether the content can be propagated. If Content is intended for Consumption, it assesses the Viewing Propagation Information. If Content is intended for Storage, the Source assesses Movement and Copying Propagation Information. In both cases, if the requested action is permitted, Content may be propagated.

If Content is intended for Export or Processing, it assesses both. If at least one is permitted, Content can be propagated.

If the Source CPCM Instance is a Processing Entity, the Source assesses only the actions that were permitted when receiving the Content Licence.

If Content can be propagated, the *CPCM_put_CL_message* (that may also be embedded in the Content), indicates which actions are authorized (i.e. Movement and Copying and/or Viewing).

Content may be propagated from the Source to the Sink if and only if one or more of the following USI indicates an allowed usage at the Sink Movement:

- MLAD/MGAD/MAD/MCPCM or VLAD/VGAD/VAD/VCPCM.

- MLocal or VLocal.

Corresponding tests can be made in any order.

If Content propagation is restricted to the Local Environment (MLocal and/or VLocal, MLAD and/or VLAD) and the Content item is to traverse multiple inter-CPCM Device connections in the home network, then proximity tests shall be performed for inter-Device connection so that end-to-end Localization of the Content transfer is assured.

7.5.2.3.1 MLAD and VLAD

If any of the RAR rules has expired or if no RAR rule is set, Content is considered to be MAD or VAD. The MLAD/VLAD restriction no longer applies.

Else, before sending MLAD or VLAD Content, the Source CPCM Instance shall perform a proximity test with the Sink CPCM Instance. If the test fails, Content shall not be sent unless permitted by other propagation USI. MLAD Content (regardless of its Viewing Propagation Information) may be Moved to a non-CPCM device. In that case, it shall add its CIC identifier in field last_CL_issuer.

NOTE: The addition of CIC identifier can be performed systematically in this case. If the Content is directly encrypted / signed using AD secret, this allows to push the Content to another Device without needing to check whether the Sink is a CPCM Instance or not.

7.5.2.3.2 MGAD and VGAD

If one of the RAR rules has expired or if no RAR rule is set, Content is considered to be MAD or VAD. The MLAD/VLAD restriction no longer applies.

No specific control needs to be done before sending the Content. Content can be Moved to a non-CPCM device if marked MGAD (regardless of its Viewing Propagation Information); in such case, the sending CPCM Instance shall add its CIC identifier field last_CL_issuer.

7.5.2.3.3 MAD and VAD

No specific controls needs to be done. Content may be sent to non-CPCM devices if marked MAD (regardless of its Viewing Propagation Information).

7.5.2.3.4 MCPCM and VCPCM

No specific controls needs to be done. Content can be sent to non-CPCM devices if marked MCPCM (regardless of its Viewing Propagation information).

7.5.2.3.5 MLocal and VLocal

If Content is not marked MLocal or VLocal, it may not be Moved (or Viewed) in the proximity, unless authorized by other propagation USI. The Source CPCM Instance may not send Content based on this propagation USI.

Before sending Content Marked MLocal or VLocal, the sending CPCM Instance shall perform a proximity test with the receiving CPCM Instance. If it fails, the Content shall not be sent unless permitted by other propagation USI. MLocal or VLocal Content may only be sent to CPCM Instances.

If an Acquisition Point or a Storage Entity sends pulled Content to a Processing Entity, it shall check as well that the final CPCM destination (i.e. a CP, an EP or a SE) is in proximity as well. This is achieved by performing the proximity test with the CPCM Instance indicated as being the initial Source or the final Sink of the Content transfer in the corresponding ("get") Content Licence exchange protocol message. A Processing Entity sending Content is not required to perform additional checking.

NOTE: A process similar to the one described in clause 7.5.2.3.1 could be used to enable Local Movement through a bit bucket. However, bit bucket is only possible when using AD protection for the Content Licence. Content stored on the bit bucket will thus only be consumable by CPCM Instances belonging to the same AD. The Movement would thus have been authorized by the MLAD/MGAD/MAD/MCPCM USI.

7.5.2.4 Export/Output Control

Export/Output controls do not require specific verifications before sending the Content. Content may be sent to CPCM Instances and to non-CPCM devices.

7.5.2.5 Ancillary Control

If the CPCM Content is marked DNCS, the CPCM Instance shall not scramble the CPCM Content before sending it. Content may be transmitted to CPCM Instances or non-CPCM devices.

If the CPCM Content is NOT marked DNCS and Content Licence is sent under the sole protection of SAC key (i.e. not under the direct protection of AD Secret, see clause 8.6.1), the CPCM Instance shall first verify the Sink CPCM Instance is LSA capable. If not, Content item may not be sent.

7.5.3 Controls to be enforced when receiving Content

A CPCM Instance shall not handle received Content unless all the following controls of this clause and its subclauses allow such action for such Content. For avoidance of doubt, unless explicitly stated, any disallowance of an action below per one USI field overrides all other permissions for such action that might be allowed per other USI fields.

7.5.3.1 Copy and Movement Control

7.5.3.1.1 Copy Control Not Asserted

No specific control is required. A Storage Entity may generate multiple copies of such Content.

7.5.3.1.2 Copy Once

A Storage Entity receiving Copy Once Content shall replace the former Content Licence with a Content Licence with Copy No More as a Copy and Movement Control before storing the CPCM Content.

7.5.3.1.3 Copy No More

A Storage Entity shall not store Content marked Copy No More except as a result of performing the Move Protocol (see clause 7.7).

7.5.3.1.4 Copy Never and Zero Retention

A Storage Entity shall not Store Content marked "Copy Never - Zero Retention Asserted".

A Storage Entity may store Content marked "Copy Never - Zero Retention Not Asserted". In such case, the Storing CPCM Instance shall issue a new Content Licence with the Copy Control Information set to "Copy Never - Zero Retention Asserted" before Storing the Content. This Stored Content shall not be made available for any purpose other than maintaining the temporary Copy in that CPCM Instance. Also, this Stored Content shall be kept for no longer than the time allowed for this purpose by the C&R regime.

A CPCM Instance shall not accept CPCM Content that is marked "Copy Never - Zero Retention Not Asserted" or "Copy Never - Zero Retention Asserted" that is not Sourced by an Acquisition Point or a Processing Entity.

7.5.3.2 Consumption control

7.5.3.2.1 Viewable

A Consumption Point or an Export Point may only Consume or Export CPCM Content if it is marked Viewable. Else, Consumption or Export shall not be permitted.

This USI shall not impact the behaviour of a Storage Entity or a Processing Entity.

7.5.3.2.2 View Window Activated

A Consumption Point or an Export Point may only Consume or Export the Content if current date and time is within the view_window. If the CP or the EP is not absolute time capable, it may request time from other CPCM Instances using protocols described in TS 102 825-5 [i.8]. If it cannot get secure absolute time, consumption shall not be permitted.

7.5.3.2.3 View Period Activated

No specific control is required by a Processing or a Storage Entity receiving VPA Content.

A Consumption Point or an Export Point shall not handle VPA Content.

NOTE: A Consumption Point or an Export Point should never receive Content Marked VPA as the Source CPCM Instance should have started the View Window and issued accordingly a new Content Licence.

7.5.3.2.4 Simultaneous View Count Activated

A Processing Entity or a Storage Entity has no specific control to perform.

For a Consumption Point or an Export Point, if a CL is received via a CPCM_put_CL protocol call from an Acquisition Point and protected by a SAC, then no specific control is required.

In all other cases, the Export Point or the Consumption Point refers to the CLC field in the CL and checks whether the corresponding Acquisition Point is available or not. If not, it may not handle the Content.

Else, the Consumption Point or Export Point builds a SAC with the Acquisition Point and requests the permission through the message *CPCM_get_permission_message* and waits for the response, then responds as follows:

- *CPCM_get_permission_response_message* with a positive or a negative answer. If answer is positive, no further control is required. If it is negative, the Export Point or the Consumption Point may not handle the Content.
- *CPCM_put_CL message* from an Acquisition Point and SAC protected. In that case, no further control is required.
- *CPCM_put_CL_message*. In that case, a new CL has been issued and the CL shall be analysed and enforced as such.

When handling a Content marked SVCA, a Consumption Point or an Export Point shall be ready to answer to *CPCM_get_content_item_status* messages. To that end, the Export Point or the Consumption Point checks whether it is still consuming/exporting the Content corresponding to the CLID included in the message. If so, the Export Point or the Consumption Point shall answer positively, else it may answer negatively or not answer at all.

7.5.3.3 Propagation control

The *CPCM_put_CL_message* indicates which action (i.e. Movement and Copying and/or Viewing) may be performed with the Content.

A Processing Entity may process the Content but should assess beforehand whether Content transmission will be permitted.

The behaviour of an Export Point will be determined by the C&R regime regarding the permitted actions and the nature of the Copy Protection system to which Content is about to be Exported.

7.5.3.3.1 MLAD and VLAD

If any of the RAR rules has expired or if no RAR rule is set, Content is considered to be MAD or VAD. MLAD/VLAD restriction no longer applies.

If Content is received from a CPCM Instance, no specific control is required.

If Content is received from a bit bucket, Content is accepted only if the CPCM Instance that Moved the Content to the bit bucket (whose CIC identifier shall be present in field last_CL_issuer) is in Proximity. If no CIC identifier is present in the auxiliary data, then the Content shall not be accepted.

If the Content Licence is directly protected by the AD secret (i.e. not under SAC protection), the receiving CPCM Instance can not know the origin of the Content and shall thus behave as if it were coming from a bit bucket.

7.5.3.3.2 MGAD and VGAD

If any of the RAR rules has expired or if no RAR rule is set, Content is considered to be MAD or VAD. MLAD/VLAD restriction no longer applies.

The Sink CPCM Instance shall perform the following tests in the order shown:

- Perform a proximity test with the Source CPCM Instance (or with the CPCM Instance indicated in the last_CL_issuer field if the Content was stored in a non-CPCM device). If the test fails (or if there is no designated CPCM Instance in the last_CL_issuer field), the Sink CPCM Instance may not handle the Content unless the next test succeeds. Else, the Sink CPCM Instance may handle the Content without performing the next test.
- If the Sink CPCM Instance is in the geographic footprint specified in CPCM auxiliary data, then it can handle the content. If the Sink CPCM Instance is not geographically aware, it can assess this using one of the protocols described in clause 6.5.8. Else, the Sink CPCM Instance may not handle the content.

In order to assess which geographic footprint is permitted, the Sink CPCM Instance shall first verify the digest of CPCM auxiliary data is equal to the one present in Content License. Else, or if CPCM auxiliary data is not available, the test fails and it may not handle the Content.

7.5.3.3.3 MAD and VAD

No specific controls are required.

7.5.3.3.4 MCPCM and VCPCM

No specific controls are required.

7.5.3.3.5 MLocal and VLocal

A CPCM Instance receiving Content marked VLocal or MLocal shall handle it only if the Content is Sourced from another Local CPCM Instance, possibly with intermediate handling by one or more Local Processing Entities.

If a Consumption Point, Export Point, or a Storage Entity receives pushed Content from a Processing Entity, it shall check as well that the initial CPCM Source (i.e. an AP or SE) is in proximity as well. This is achieved by performing the proximity test with the CPCM Instance indicated as being the initial Source of the Content transfer in the corresponding ("put") Content Licence exchange protocol message. A Processing Entity sending Content is not required to perform additional checking.

NOTE: A process similar to the one described in clause 7.5.3.3.1 could be used to enable Local Movement through a bit bucket. However, bit bucket is only possible when using AD protection for the Content Licence. Content stored on the bit bucket will thus only be consumable by CPCM Instances belonging to the same AD. The Movement would thus have been authorized by the MLAD/MGAD/MAD/MCPCM USI.

7.5.3.4 Export and Consumption output control

Only Export and Consumption Points may perform these controls.

A Consumption Point shall perform the following:

- If Content is intended to be Consumed through a Controlled CPS, the Export/Output Controlled CPS and relevant flag in the Controlled CPS Vector must be asserted. Else, consumption shall not be permitted.
- If Content is intended to be Consumed in analogue standard definition and disable_Analogue_SD_Consumption is asserted, consumption shall not be permitted.
- If Content is intended to be Consumed in analogue high definition and disable_Analogue_HD_Consumption is asserted, consumption shall not be permitted.
- If Content is intended to be Consumed through the means of an Analogue high definition output and image_constraint is set, relevant processing described in TS 102 825-3 [i.11] shall be performed before Consuming the Content.

An Export Point shall:

- If Content is intended to be Exported to a Controlled CPS, the Export/Output Controlled CPS and relevant flag in the Controlled CPS Vector must be asserted. Else, export shall not be permitted.
- If Content is intended to be Exported to an Untrusted Space and Export_beyond_trust is not asserted, then export shall not be permitted.
- If Content is intended to be exported in analogue standard definition and disable_analogue_SD_Export is asserted, export shall not be permitted.
- If Content is intended to be Exported in analogue high definition and disable_analogue_HD_Export is asserted, export shall not be permitted.
- If Content is intended to be Exported to an Untrusted Space and image_constraint is set, relevant processing described in TS 102 825-3 [i.11] shall be performed before Exporting the Content.

7.5.3.5 Ancillary control

If the CPCM Content is marked DNCS, the CPCM Instance does not need to descramble the content before handling it.

7.5.4 Remote Access rules

7.5.4.1 General

Remote Access Rules may be verified before sending Content or after reception depending upon which CPCM Instance needs to assess whether Content can be remotely accessed.

As soon as one of the Remote Access Rules has been satisfied or if none Remote Access Rules is asserted, Content shall be treated as if it were marked MAD or VAD.

7.5.4.2 Remote Access post record

The "Remote Access post record" restriction expires as soon as the service event from which the Content item originated ends. These controls are exclusively performed by the non-CPCM portion of a product that contains the CPCM Acquisition Point from which the Content item was Acquired. Upon service event completion as determined through non-CPCM specified methods, the Acquisition Point shall issue a new Content Licence and transmit it to Storage Entities to which it had transmitted the Content Licence (possibly through a Processing Entity). The new Content Licence shall have the three Remote Access Rules flags unasserted, Movement and Copying Propagation Information shall be set to MAD and Viewing Propagation Information shall be set to VAD.

At any time, the Storage Entity may ask the Acquisition Point (known from the CLC field) whether the service event has ended or not using the CPCM_get_permission protocol.

Upon receiving message CPCM_get_permission for a Remote Access Post Record content, an Acquisition Point shall issue a new Content Licence if the corresponding service event is finished. Before issuing the new CL, it shall clear the three Remote Access Rules flags and set Movement and Copying Propagation Information to MAD and Viewing Propagation Information to VAD. Else, it refuses the permission.

It sends then the new Content Licence using *CPCM_put_CL_message*.

7.5.4.3 Remote Access post date/time (moving window)

The "Remote Access post date/time (moving window)" restriction starts expiring when the current date and time equals the date and time specified in this field. To assess this, CPCM Instance shall have access to Secure Absolute Time.

Once the "Remote Access post date/time (moving window)" restriction starts expiring, a given instant of the Content is available for Remote Access if current date and time is later than the date specified of this field augmented by the time elapsed since the beginning of the Content. The Content will be fully available Remotely only if the current date and time is later than this field augmented with the total Content duration.

EXAMPLE: If Content is marked to allow remote access 24 hours after transmission, then the Acquisition Point would note the date/time that the Content item was first Acquired (when the recording started and the Content Licence was created) and add 24 hours to that time for inclusion within the Content Licence. The playback could commence upon the date/time stored in the CL, but fast forwarding is not to be allowed to a point in the Content item that has not been delayed for 24 hours.

7.5.4.4 Remote Access post date/time (immediate)

The "Remote Access post date/time (immediate)" restriction is expired if the current date and time is later than the value specified in this field. To assess this, CPCM Instance shall have access to Secure Absolute Time.

Once the current date time is over, the whole Content shall be available for Remote Access, without any further restriction.

7.6 CPCM Content Scrambling management

7.6.1 General

CPCM Content shall be scrambled using the CPCM Local Scrambling Algorithm (LSA) (see TS 102 825-5 [i.8]) unless it is marked DNCS, in which case it shall not be scrambled using the CPCM LSA.

The CPCM LSA has two possible chaining modes and two possible IV preparation modes, thus resulting in four possible combinations of these scrambling options.

The CPCM Instance acting as CPCM Content Source decides which of these four combinations to apply for any CPCM Content item (or protected CPCM Content stream).

Depending on its implementation and the requirements of he relevant C&R regime, the CPCM Instance could also always apply the same combination to all CPCM Content that it Sources, hence a CPCM Instance that can only Source CPCM Content does not necessarily need to implement the other CPCM Scrambler combinations.

In order to guarantee interoperability with all potential Sources of CPCM Content, all Sink CPCM Instances shall implement all four combinations of the CPCM Descrambler options.

7.6.2 Dynamic changes of the CPCM Content scrambling key

This clause provides the generic mechanism whereby a CPCM Instance acting as a Source of a scrambled CPCM Content item is able to force a change of the content scrambling key. The application of this generic mechanism to particular CPCM Content formats is defined in TS 102 825-9 [i.3].

There is no mechanism of crypto-periods built into the CPCM System content management scheme, but a mechanism is specified whereby a Source (CPCM Instance) of a CPCM Content item can force a change of the CPCM Content encryption key and ensure the correct synchronization of the change in the Sink (CPCM Instance). Thus crypto-periods can in effect be applied by the Source CPCM Instance, if necessary, by repeated application of the process to force a change of the content scrambling key, as specified for the Acquisition Point (see clause 7.4.1).

It is not specified when, or how often, such key changes shall occur, so it is left up to the Source CPCM Instance implementation as to how often, or whether at all, such dynamic content encryption key changes are applied.

Figure 39 shows the generic process for the changing of the CPCM Content scrambling key, depicting two cycles of CPCM Content scrambling key changes.

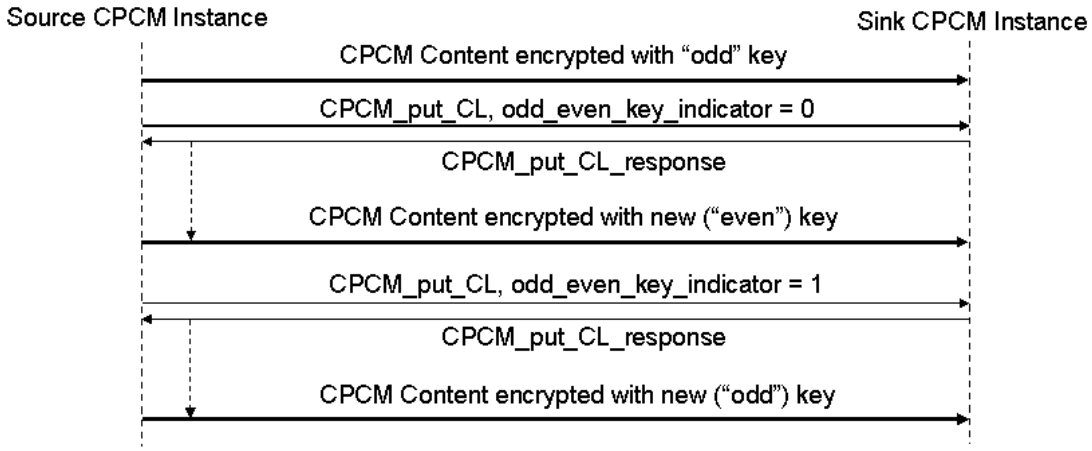

Figure 39: Sequence diagram for changing the CPCM Content scrambling key

The scrambling_information field of the Content Licence contains the odd_even_indicator bit that is set by default to "0", indicating nominally the application of the odd CPCM scrambling key.

When the Source CPCM Instance intends to change the CPCM scrambling key applied to the Content item being streamed to a Sink CPCM Instance, it shall issue in advance a new Content Licence to the Sink CPCM Instance, using the CPCM_put_CL_message. The new Content Licence shall contain the odd_even_indicator bit toggled from the previous setting, i.e. for the first content key change it shall be set to "1", indicating the use of the even CPCM scrambling key. The new Content Licence shall also contain the new CPCM descrambling key chosen at random by the Acquisition Point.

The change of content scrambling key must be synchronized with the Sink CPCM Instance. How this is done depends on the CPCM Content format deployed. This is one aspect defined in the CPCM adaptation layer for CPCM Content formats in TS 102 825-9 [i.3].

Due to the fact that only one CPCM Content scrambling key can be stored in each CPCM Content Licence, the implication of such key changes when the associated CPCM Content item is being Stored by a Sink CPCM Instance is that one of the following actions shall be undertaken:

- The Source CPCM Instance shall inhibit such content key changes when that CPCM Content item is being Stored by a Sink CPCM Instance.
- The Sink CPCM Instance shall decrypt each segment and re-encrypt all segments to reconstruct the original CPCM Content item using a newly generated CPCM Content scrambling key.
- The Sink CPCM Instance shall create a new CPCM Content item for each segment of the original CPCM Content item for which a new content key is applied. It is possible to maintain a logical link between the resulting set of segments by appropriate handling of CLID for each segment. The C&R regime might require that this procedure is applied to Content items that exceed a certain length.

7.7 Content Move operation

7.7.1 Introduction

The CPCM Content Move protocol shall be used for all cases where a CPCM Content item is Moved between CPCM Devices.

The Content Move Protocol actually consists in Content Licence Moving. Encrypted Content may actually be copied without any restriction and without violating the USI as long as one single instance of the Content Licence exists. The Content may be copied either before or after the Content Licence Move.

The Content Licence Move protocol, shown in figure 40, is realized using a transactional CPCM protocol (see clause 6.2.3).

The CPCM Content Move Protocol may be used to Move Content marked Copy No More or Content marked VPA before the first Playback or the first Copy occurs.

The Content Move operation may traverse one or more Processing Entities hosted in other CPCM Instances. In such cases, the Content Licence shall be Moved from the Source CPCM Instance to the Sink CPCM Instance hosting the Processing Entity. The Sink CPCM Instance processes the Content Licence, obtains the descrambling key, possibly generates a new descrambling key (see clause 7.4.2) and proceeds then to the Content Licence Move to the next Processing Entity or to the final Sink CPCM Instance. It retains the descrambling key(s) and uses it to process the Content. Once Processing is completed, it shall discard the descrambling key(s).

If the Sink CPCM Instance is Remote and the Source and/or Sink CPCM Instance is not capable of Remote Access, the Content Licence Move protocol may be run through Local Master capable Instances. In such case, Content Licence shall be Moved successively between each of the involved Instances.

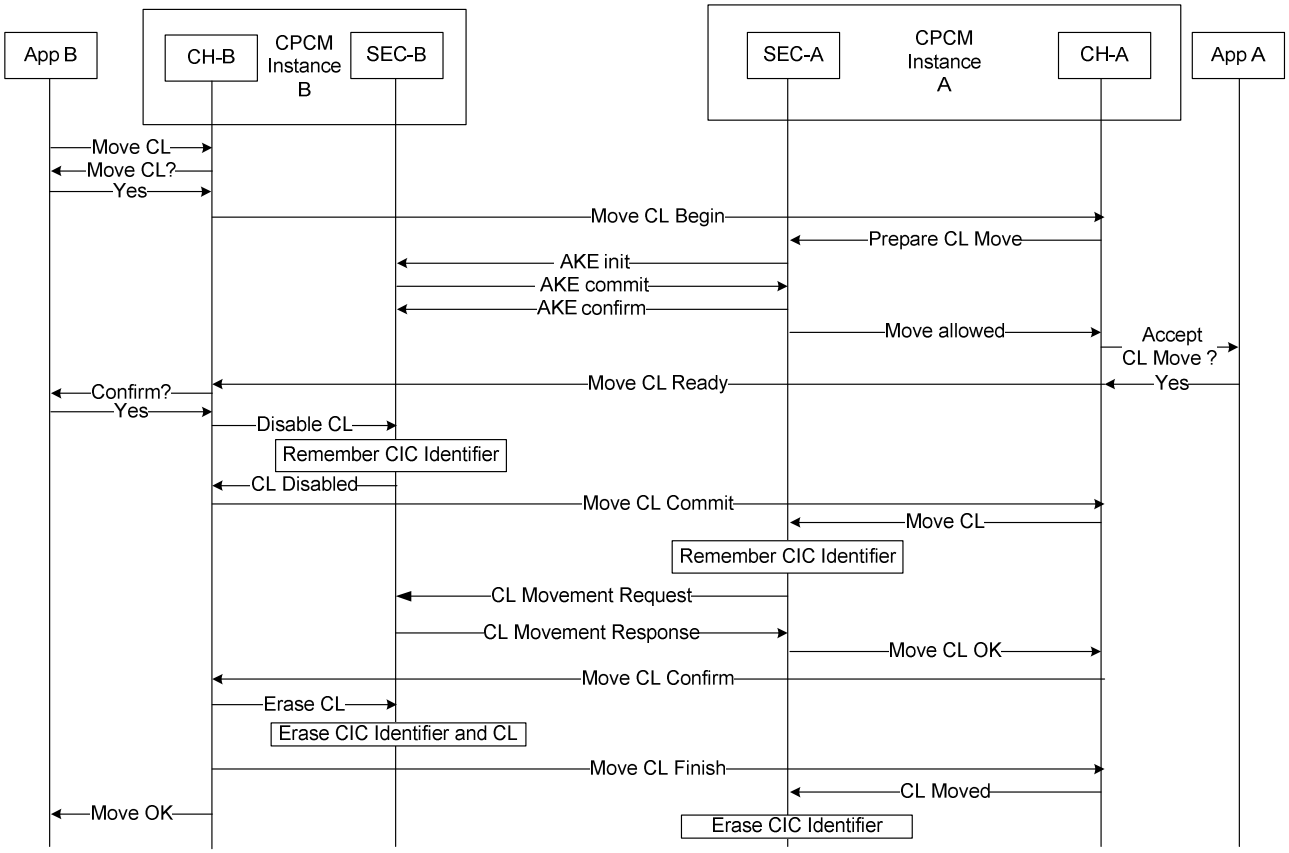

Figure 40: Content Move protocol

7.7.2 Source CPCM Instance behaviour with Content Move

7.7.2.1 Content Handling Behaviour

A CPCM Instance that needs to Move a Content Licence to another CPCM Instance shall behave as depicted in figure 41:

- First, the Device Application is asked to confirm that it wishes to Move the Content Licence.
- Assuming the confirmation succeeds, the Source CPCM Instance sends a *move_CL_begin_message* to the CPCM Instance to which Content Licence is intended.
- If no *move_CL_ready_message* response is received within timeout T1, or if a transaction_rollback_message occurs, the Source CPCM Instance stops the protocol.
- Next, the Source CPCM Instance shall ask the Device Application to confirm that it still wishes to Move the CL.

- If the Device Application confirms the CL Move, the Source CPCM Instance checks with its Security Control whether it can proceed. If so, then a *move_CL_commit_message* shall be issued. If the Device Application or the Security Control refuses the Move, the Source CPCM Instance shall send a Secured transaction_rollback_message.

- If no *move_CL_confirm_message* is received within timeout T1 then it shall stop the protocol.

Finally the Source CPCM Instance shall inform its Security Control that the Move has been completed and issue a *move_CL_finish_message*.

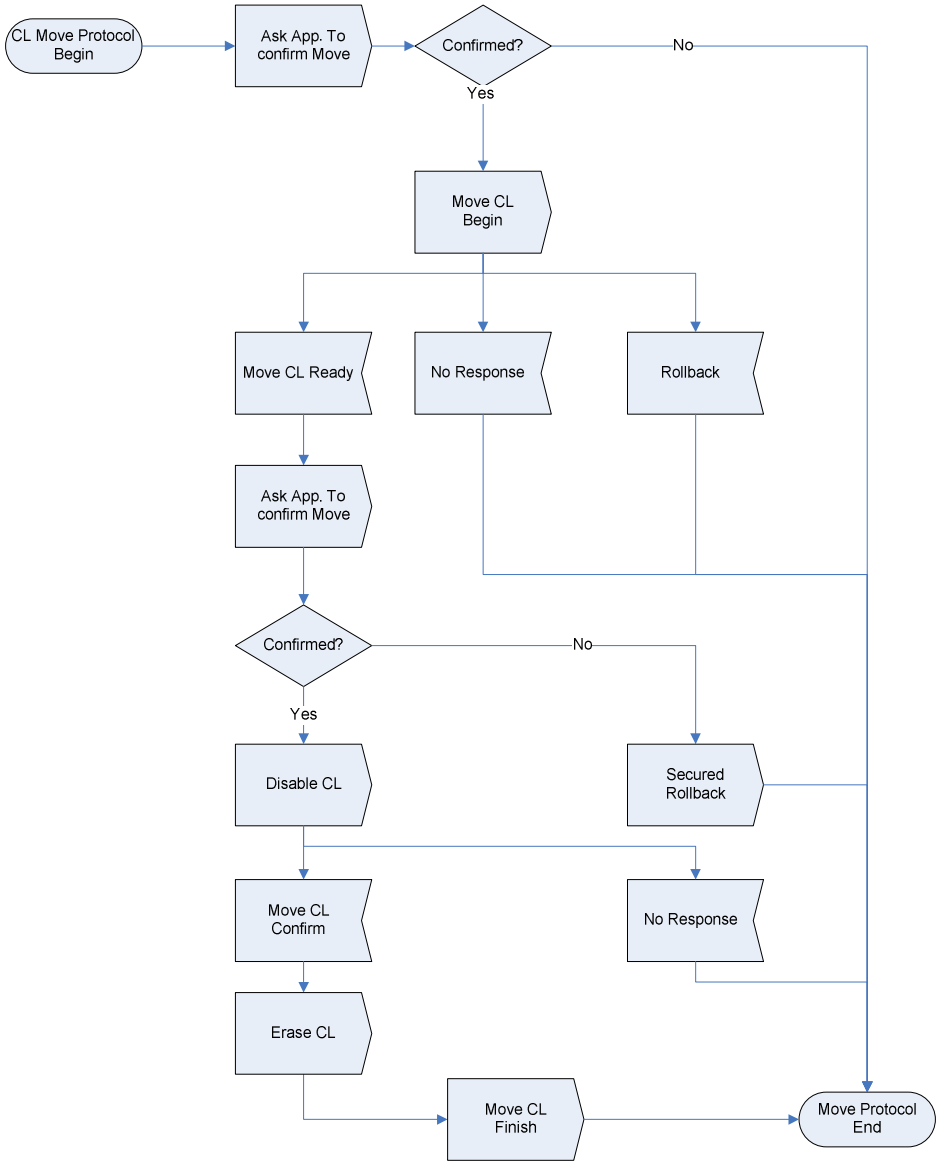

Figure 41: Source CPCM Instance in Content Move protocol

7.7.2.2 Security control behaviour

The Security Control is involved at four steps during the CL Move Protocol which shall be executed in the following order. Any inversion or omission shall result in a negative result and any failure of one step will cause the process to stop.

1) After the *move_CL_begin_message* is sent, it will accept the challenge to run the SAC establishment procedure the destination Instance.

2) Just before sending the *move_CL_commit_message* , the Content Handling informs its Security Control that the transfer is about to occur through a *CPCM_disable_CL_message*. Security Control checks the SAC has been established with the Destination CPCM Instance and records the CPCM Instance Certificate Identifier. It disables the Content Licence (i.e. it will no longer use the Content Licence).

3) Upon receipt of a *CL Movement request* message, Security Control sends the Content Licence through a *move_CL_response_message*.

4) Once the Content Handling has received a *move_CL_confirm_message*, it requests its Security Component to erase the CL. The Security Control erases both the CL and the recorded CPCM Instance Certificate Identifier.

If the protocol is intentionally interrupted by the Device Application between second and fourth step (e.g. because the user decided to run the protocol with another Domain Controller), all the recorded data (CL and CPCM Instance Certificate Identifier) shall be erased.

7.7.3 Sink CPCM Instance behaviour with Content Move

7.7.3.1 Content handling behaviour

In order for a CPCM Instance to receive a CL through a Move protocol, the Source CPCM Instance shall perform the following tasks as depicted in figure 42.

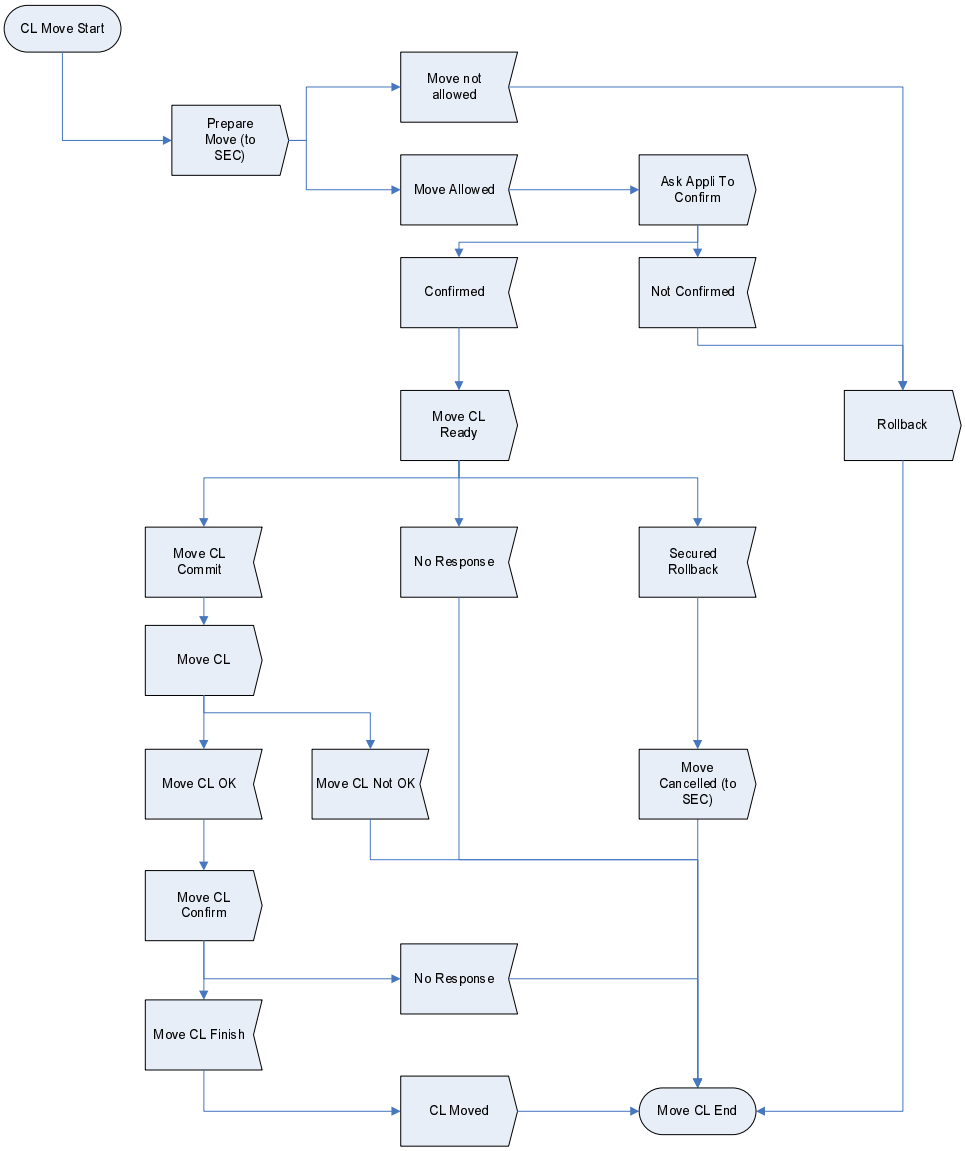

Figure 42: Destination CPCM Instance in Content Move protocol

When receiving a Move CL request, the Source CPCM Instance shall do the following.

- Checks with its Security Control as to whether the Move is possible or not. If not, it informs the requesting CPCM Instance through a *transaction_rollback_message* and terminate the process.
- Request the Device Application to confirm the Move. If the Device Application does not confirm the Move, then it informs the requesting CPCM Instance through a *transaction_rollback_message* and terminate the process.
- Send a *move_CL_ready_message* to the requesting CPCM Instance to indicate that the CL Move may happen.
- It waits for *move_CL_commit_message* from the requesting Instance. If no *move_CL_commit_message* is received within timeout T1, terminate the process. If a Secured *transaction_rollback_message* is received, the Domain Controller informs its Security Control that the transfer has been cancelled.
- Having received the *move_CL_commit_message*, it requests its Security Control to proceed with the CL Move.
- If the Security Control reports that the transfer did not occur properly, terminate the process.
- Else, send a *move_CL_confirm_message*to the requesting CPCM Instance which responds with a *move_CL_finish_message*.
- It informs the Security Control that the Move has been completed.

7.7.3.2 Security control behaviour

The Security control is active in three stages. These steps shall be conducted in the order described below, if there is a failure at any stage it shall be reported as a protocol error and the process shall end.

1) The Content Handling requests its Security Control to prepare for the Move. To that end, the Security Control then runs the SAC establishment procedure with the Security Control of the requesting CPCM Instance. During SAC establishment, it verifies that all relevant information transmitted within the request match the Certificate information. If anything fails, Security Control signals this to Content Handling.

2) Upon reception of a *move_CL_commit_message*, Content Handling informs its Security Control to proceed with the Move.

   Then, the Security Control records the requesting CPCM Instance Certificate identifier. It requests then the Content Licence in a *CL_movement_request_message* and waits for response. Once the *CL_movement_response_message* is received, it records the Content Licence and informs its Content Handling that the Move was successful. If this message was not received within T1, it informs its Content Handling of the Move failure.

   If the Device Application intentionally interrupts the protocol at that step (e.g. because it knows it will never finish), the CPCM Instance Certificate Identifier shall be erased.

3) Finally, upon receipt of a *move_CL_finish_message*, or upon other events as described in clause 7.7.5, the Content Handling informs its Security Control that the Move was achieved. Security Control then erases the recorded CPCM Instance Certificate Identifier.

7.7.4 Abnormal behaviour

In the normal course of events, assuming that no CPCM protocol message time-out occurs, when a requesting CPCM Instance sends a message to another CPCM Instance the responding CPCM Instance shall always respond with a message type corresponding to that of the request. Many possible errors are not taken into account: e.g. message authentication checking may fail or the receiving Instance may be busy.

The failure of the transaction shall be signalled by the responding CPCM Instance by the sending of an error message identifying the cause of the error (see clause 6.3.3).

Other abnormal behaviour occurs when a message is received that is not expected at that stage, or when the Control Handling of the CPCM Instance receives a *move_CL_confirm_message*when its Security Control had not previously received the relevant message. In these cases, a *CPCM_protocol_error_message* (if no SAC has been established with the sending CPCM Instance) or a Secured *CPCM_protocol_error_message* (if a SAC already exists with the sending CPCM Instance) is sent.

When receiving an error message, depending on the nature on the error, the CPCM Instance may decide to retry to send the previous message or to abort the protocol. Re-sending shall only occur a limited number of times. Aborting a protocol will result in the same effect as if no response was received.

When no response to the previous message is received, implementation may decide also to retry to send the previous message before considering that no response will be received.

Upon receipt of a *transaction_rollback_message* or Secured *transaction_rollback_message*, a CPCM Instance shall stop running the current protocol and revert to the state it was before running the protocol returning any reserved resources to their original state. If it wishes to retry to run the protocol, it must re-start it from the beginning.

7.7.5 Failure recovery

The Content Move protocol has been designed to be able to restart and finish successfully if it is unexpectedly interrupted (e.g. because of a power failure or a network problem that makes one of the two peers unreachable).

NOTE: There is a risk that CL may be lost during and after the interruption if the Instances suffer loss of state information prior to re-connection, or are never re-connected.

The following describes the normative behaviour of a CPCM Instance with a stored CPCM Instance Certificate Identifier. The behaviour supposes that the user waits for the client and the server to be re-connected after the interruption.

At any time, the Device Application may force the client to restart the protocol with another server or tell the server that there is no longer any need to remember the client identifier as the protocol will not restart. This feature is useful if a CPCM Instance can only remember a few CPCM Instance Certificate identifiers and it is therefore able erase one of these in order to be able to run another protocol. The user shall be made aware that this may cause undue loss of Content Licence.

When a Source CPCM Instance with a stored CPCM Instance Certificate Identifier discovers the corresponding CPCM Instance, it shall re-initiate the Move Protocol using a *move_CL_begin_message*.

A Destination CPCM Instance with a stored CPCM Instance Certificate Identifier may not Move the Content Licence to another CPCM Instance.

A Destination CPCM Instance with a stored CPCM Instance Certificate Identifier accepts any restart of the protocol by the relevant CPCM Instance. It erases the CPCM Instance of the Identifier if the relevant CPCM Instance is discovered but it does not receive any *move_CL_begin_message* within T2.

7.8 CPCM Content Revocation

7.8.1 General

The CPCM System includes two distinct modes of CPCM Content-based revocation - per CPCM Instance, and per AD. CPCM Content revocation is performed by storing the CIC identifiers of the revoked CPCM Instances, and/or the ADS digest of the revoked ADs, for a particular CPCM Content item, within a CPCM Revocation List, the structure of which is specified in clause 5.4.8.

The actual method of delivery of CPCM revocation lists is out of scope for the CPCM System specification, but the following principle methods for their delivery are foreseen:

- To an Acquisition Point, embedded inside an Input Content stream (e.g. a broadcast transmission), or file (e.g. from a DRM system); or
- To a Storage Entity, embedded within a CPCM Content item stored in the DVB File Format.

Revocation list delivery methods may also be part of a CPCM System adaptation, contained in TS 102 825-9 [i.3].

The CPCM Instance hosting an Acquisition Point shall compile the complete CPCM revocation list received within its content delivery channel(s) and maintain the latest version in accordance with detailed requirements laid out by the C&R regime. The method to identify the latest version is defined in TS 102 825-5 [i.8].

After such compilation of the CPCM revocation list, there are three methods whereby CPCM revocation lists can propagate within the CPCM System:

- by being embedded in the CPCM Content stream between two CPCM Instances, similarly to how it may be carried in the Input Content delivery channel; or
- by being Stored within a file created according to the DVB File Format specification; or
- by a transfer within the home network ecosystem, initiated with the CPCM protocol, specified in xxx. The actual transfer of the CPCM Revocation List is not accomplished using CPCM, rather it is handled in the home network ecosystem. This is one aspect that shall be included in any home network ecosystem adaptation contained in TS 102 825-9 [i.3];
- by broadcasting *get_CPCM_RL_message* that requests one or more Revocation Lists (issued by different C&R regimes) with a minimum index. Each CPCM Instance having a Revocation List for the relevant C&R regime with an index not lower than the requested one shall provide the Revocation List using the *notify_CPCM_RL_message*. If a CPCM Instance has more than one of the requested lists, it shall use several instances of *notify_CPCM_RL_message*.

Each CPCM Instance shall record permanently the latest Revocation List (i.e. the Revocation List with higher index) for each of the C&R regime it implements (i.e. each bit asserted in the C_and_R_regime_mask in its CIC).

7.8.2 CPCM Instance-based Content revocation

The CPCM Content Licence structure (see clause 5.4.4) contains the field C_and_R_regime_mask, which is an 8-bit mask that also implies for which of the eight possible C&R regimes a revocation check is to be performed for the associated CPCM Content item.

CPCM Device-based content revocation checks can be applied in the following cases:

- by the CPCM Instance that hosts the Acquisition Point for Input Content against which that CPCM Instance should be checked; or
- by a Source CPCM Instance prior to delivering CPCM Content to a Sink CPCM Instance to be checked.

In order to be able to manage, use or otherwise handle CPCM Content, a CPCM Instance shall not be revoked for at least one of the C&R regimes required in the CL that it implements.

To assess whether a CIC is revoked for any given C&R regime, a CPCM Instance shall first obtain a Revocation List issued by that C&R regime whose index is higher or equal to the one given in the CL. If it does not have such a list, it may try to get it by other methods described in clause 7.8.1. If it can not obtain a relevant Revocation List, it shall not handle the content.

Else, it shall check the revocation status of ALL certificates in its CPCM Certificate chain as follows:

- If required index in the Content License is 0, CPCM Instance is not revoked. In such case, there is no need to request the corresponding Revocation List as it is empty.
- If the certificate_generation_index is lower than the minimal generation index requested in the Revocation List, and if the index requested in the Content License is not lower that certificate_generation_index_change, it is revoked and shall not handle the Content.
- If the CIC Identifier is not included in the Revocation List, it is not revoked.
- If the CIC Identifier is included in the Revocation List with a revocation index greater than the one required in the Content License, it is not revoked.

- If the CIC Identifier is included in the Revocation List with a revocation index equal to or lower than the one required in the Content License, it is revoked and shall not handle the content.

7.8.3 AD based Content revocation

An AD may be revoked if its associated AD Secret becomes compromised and could therefore be misused to make CPCM Content bound to that AD usable outside the envisaged confines of the AD.

An AD to be revoked is identified in the CPCM Revocation List by the digest (computed using Hash function described in TS 102 825-5 [i.8]) of its AD Secret.

The effect of AD revocation is that no new CPCM Content shall be bound to the revoked AD. CPCM Content that had already been Acquired and bound to the AD shall continue to be usable within that revoked AD.

If an AD is revoked then no CPCM Content shall be Acquired into that AD, regardless of the revocation status for that Content item of the CPCM Instances that are members of that AD.

AD-based content revocation checks can be applied in the following cases:

- by the CPCM Instance that hosts the Acquisition Point for Input Content against which the AD of which that CPCM Instance is a member should be checked; or

- by a Sink CPCM Instance for CPCM Content that was not able to be bound to the AD by the Source CPCM Instance.

In order to be able to protect Content Licence with AD protection or to verify the CL is correctly protected with AD secret, the CPCM Instance shall first verify the corresponding AD is not revoked for at least one of the C&R regime required in the CL.

To assess whether AD is revoked for one given C&R regime, a CPCM Instance shall first get a Revocation List issued by this C&R regime whose index is higher or equal to the one given in the CL. If it does not have such a list, it may try to get it by other methods described in clause 7.8.1. If it can not obtain a relevant Revocation List, it shall not handle the content.

Else, it verifies its revocation status as follows:

- If required index is 0, CPCM Instance is not revoked. In such case, there is no need to request the corresponding Revocation List as it is empty.

- If the ADID is not included in the Revocation List, it is not revoked.

- If the ADID is included in the Revocation List with a revocation index greater than the one required in the Content Licence, it is not revoked.

- If the ADID is included in the Revocation List with a revocation index equal to or lower than the one required in the Content Licence, it is revoked and shall not handle the Content.

CPCM Content that was bound to an AD that became revoked may be re-bound to a new AD. The process of re-binding the CPCM Content could be performed either:

- for CPCM Content that is marked MLocal, CPCM Devices can autonomously re-bind the CPCM Content to the new AD, or

- for CPCM Content not marked MLocal, the re-binding may be performed by or in conjunction with an ADMAAA. This is performed using the CPCM_get_new_CL protocol, defined in clause 6.5.3.4.

7.9 CPCM Content interoperability between C&R regimes

CPCM Instances shall not handle CPCM Content that is not covered by a CPCM Compliance and Robustness (C&R) regime that is supported by that CPCM Instance. Individual C&R regimes may allow the inclusion of coverage of CPCM Content from other C&R regimes but this is out of the scope of the CPCM specification and, if applicable, shall be specified under mutual agreement of the relevant C&R regimes.

If the C&R regime mask in the Content Licence (CL) is set to 0x00, this shall imply that that CPCM Content shall not be usable by any CPCM Instance. Such CLs shall be maintained privately and the associated content shall not be advertised within the CPCM System. Conversely, if the C&R regime mask in the CL is set to 0xFF this shall ensure that that CPCM Content gains maximum interoperability in the CPCM System.

If two C&R regimes have the ability to establish trust, they must also establish a means of arbitrating between different requirements such as ADSE and Proximity test constraints and requirements.

EXAMPLE: If one requires a certain proximity test and the other does not, the compliant treatment of proximity determination between Devices that support both C&R regimes must be defined.

8 CPCM Content Licence management

8.1 General

This clause specifies how CPCM Content Licences (CL) are managed throughout the lifetime of the associated Content items within the CPCM System.

A Content Licence is generated when a Content item is Acquired, but a new CL can also be generated upon Storage or Processing of an existing Content item. CL generation aspects are specified in clause 8.2.

Clause 8.3 is devoted to the methods of CL identification that ensure that any CL generated can be guaranteed to be uniquely identifiable within the CPCM System.

Clause 8.4 elaborates the particular issues with the verification of CLs during transfer of an associated CPCM Content item.

Clause 8.5 specifies the mechanism whereby a new version of a CL can be obtained from the original provider of the associated Content item (outside the CPCM System), for example to obtain additional Authorized Usage for that Content item.

Clause 8.6 specifies aspects concerning the protection of Content Licences within the CPCM System.

8.2 CPCM Content Licence generation

8.2.1 CL generation upon Content Acquisition

8.2.1.1 General

A unique Content Licence shall be created upon Acquisition of each CPCM Content item from a Trusted Source in the CPCM System or upon Re-Acquisition of a Content Licence (see clause 8.5).

To be able to generate a Content Licence, the CIC of the CPCM Instance hosting the Acquisition Point shall not have expired.

The CPCM Instance that Acquires the Content item shall populate the fields of the new Content Licence as specified in the following sub-clauses.

The Content Licence shall then be protected as specified in clause 8.6.

8.2.1.2 CPCM version

The CPCM_version field shall be set to the same value as that stored in the CPCM Instance Certificate of the Acquisition Device. The value 0x01 shall be applied for CPCM Instances that comply with the present specification. If the Content Licence is to be provided to a CPCM Instance that implements an earlier CPCM System version, the Acquiring CPCM Instance shall adopt that CPCM Instance's value of CPCM_version and shall construct the Content Licence appropriately for that lower-version Sink CPCM Instance.

8.2.1.3 Content Licence protection

The content_licence_protection bit in the CL shall be set according to the mode of protection applied to the Content Licence by the Acquisition Point, as specified in clause 5.4.4.

8.2.1.4 Descrambler information

The odd_even_indicator bit shall initially be set to indicate that the odd key is applied, i.e. set to the value "0".

This value is changed only if the Acquisition Point applies a change of the CPCM scrambling key. The generic process for such a change of scrambling key is specified in clause 7.6.2. How this generic process is applied to particular CPCM Content formats is specified in TS 102 825-9 [i.3].

The descrambler_chaining_mode and descrambler_MSC_mode bits are set according to the choice of the implementation of the Acquisition Point (see TR 102 825-12 [i.12]). This choice could be imposed by the Input Content delivery system.

If the Content item is not to be scrambled with the CPCM Scrambler then all descrambler_information bits shall be set to zero.

8.2.1.5 Content Licence identifier

The Acquiring CPCM Instance shall allocate a Content Licence identifier (CLID) for each new Content item that it introduces into the CPCM System. The CLID shall be unique within the CPCM System, i.e. two Acquisition Points within the CPCM System shall not be able to generate the same CLID for different Content Items. However, the Acquisition Point may allocate the same CLID to more than one Content Licence, for example when the CPCM Scrambling Key is changed. The uniqueness of CLID within the CPCM System is guaranteed by the application of a CLID allocation scheme, whereby the Acquisition Point is able to maintain the uniqueness of the CLIDs it generates within that allocation scheme. The CLID allocation scheme is specified in clause 8.3.

8.2.1.6 Content Licence creator

The content_licence_creator field shall be set to the CIC Identifier of the CPCM Instance Acquiring the Content item.

The content_licence_creator field of a CL shall not be changed by any CPCM Instance after it has been set by the CPCM Instance that Acquired the Content item.

8.2.1.7 Last CL issuer

This field shall be set only when the Content Licence is protected using the AD Secret (i.e. not under the direct protection of SAC secret, see clause 8.6.1) and if needed, according to the USI settings of the Content item (see TS 102 825-10 [i.10]). Otherwise the use of this field is optional.

If used, it is set to the CIC Identifier of the CPCM Instance that is Acquiring the Content item.

8.2.1.8 C&R regime mask

The C&R regime mask field shall be set to the value as authorized by the delivery system of the associated Input Content item.

If no value is given by the delivery system, it shall be set according to the C&R regime mask field in the CIC. If the Acquisition CPCM Instance has more than one CIC then apply the logical OR of the C&R regime masks of each CIC.

If the Acquisition Point is revoked for one C&R regime and if a revocation check is required, it may not set the corresponding bit in the C&R regime mask.

8.2.1.9 RL index list

This field shall be populated for each of the C&R regimes asserted in the C&R regime mask in the following way:

- The value delivered by the delivery system, if any;
- Otherwise, the index of the most recent Revocation List known to the Acquisition Point.

8.2.1.10 Authorized Domain identifier

If the Input Content is required to be bound to an AD then the Authorized Domain identifier (ADID) field in the Content Licence shall be set to the ADID of the AD to which the Acquiring CPCM Instance belongs.

If the Acquiring CPCM Instance is not AD aware then the ADID field shall be set to zero.

If the CPCM Instance is AD aware but is not member of an AD, the Acquisition Point shall do one of the following:

- Create a new Authorized Domain and apply the ADID of the newly created AD; or
- Set the ADID field in the CL to zero.

The CPCM Implementation Guidelines (TR 102 825-12 [i.12]) explains the consequences of each option.

8.2.1.11 Descrambling key

The CPCM Content descrambling_key shall be set randomly.

If the Content item is not to be scrambled with the CPCM Scrambler then the descrambling_key shall be set to zero.

The C&R regime may set a maximum time period for the validity of a single CPCM Content descrambling key. A CPCM Content Source shall apply a different descrambling key to each CPCM Content item that it Sources, but if it Sources a Content item that is longer in duration than this maximum period the CPCM Content Source shall apply a change of descrambling key to the CPCM Content item being Sourced. This is accomplished by the procedure specified in clause 7.6.2.

8.2.1.12 Usage State Information

The Usage State Information (USI) shall be set according to the requirements of the Input Content. Such settings are totally at the discretion of the content provider on a per Content item basis.

Where standards exist for relevant content usage rules signalling in other systems, there can also be standardized mappings from these to CPCM USI. These are defined in TS 102 825-10 [i.10].

8.2.1.13 CPCM Auxiliary Data digest

This field is present only if there is any CPCM Auxiliary Data (embedded in the CPCM Content) associated with the Content item.

This field is the digest of the CPCM Auxiliary Data, computed using the hash function defined in TS 102 825-5 [i.8].

8.2.1.14 AD Secret Signature

This field is present only when Content Licence is self-protected using AD Secret.

It is computed as the Message Authentication Code (see TS 102 825-5 [i.8]) of all other fields of the CPCM Content Licence.

AD Secret signature computation is part of the Content Licence protection process (see clause 8.6).

8.2.2 CL generation in other cases

8.2.2.1 General

A Content Licence may also be generated in the following circumstances:

- When the Content Licence protection mode is changed (see clause 8.6). This can occur as a result of CPCM Storage or Processing or when the Content License is exchanged between CPCM Instances (see clause 8.6.3);

NOTE: Transitions between SAC Secret Protection and Device Secret Protection do not imply the generation of a new Content Licence.

- When a Storage Entity changes the USI (see clause 8.6) or changes the scrambling keys and modes;
- Acquisition Points may modify a Content Licence when some conditions are met (e.g. end of a recording)

Only the above modifications can be done under conditions dictated by the C&R regime; for instance, a Processing Entity may not change any of the USI.

The following subclauses detail which fields of the Content Licence may be changed. All other fields shall be kept unchanged from the original Content Licence.

8.2.2.2 Content Licence protection

The content_licence_protection bit may be changed if the CPCM Instance handling the Content Licence change the Content Licence Protection.

The settings of the content_licence_protection bit are described in clause 5.4.4.

8.2.2.3 Descrambler information

This field may be changed when a Processing Entity changes the scrambling keys or crypto-periods are suppressed from the Content item (see clause 7.6.2).

The odd_even_indicator bit shall initially be set to indicate that the odd key is applied, i.e. set to the value "0".

This value is changed only if the Acquisition Point applies a change of the CPCM scrambling key. The generic process for such a change of scrambling key is specified in clause 7.6.2. How this generic process is applied to particular CPCM Content formats is defined in TS 102 825-9 [i.3].

The descrambler_chaining_mode and descrambler_MSC_mode bits are set according to the choice of the implementation of the Acquisition Point, unless a specific mode is required by the delivery system.

If the Content item is not to be scrambled with the CPCM Scrambler then all descrambler_information bits shall be set to zero.

8.2.2.4 Content Licence identifier

This field shall be updated when creating a new Content Licence as described in clause 8.3.

8.2.2.5 Last CL issuer

This field shall be updated only when the Content Licence is transmitted under the sole protection of the AD Secret (i.e. not under the direct protection of SAC secret).

It shall be set to the value of the CIC Identifier of the CPCM Instance generating the Content Licence.

8.2.2.6 Authorized Domain identifier

This field shall be updated under two circumstances as described in clause 8.6.3:

- When the Content item becomes bound to a different AD. In this case ADID shall be set to the value of the ADID of the new AD.
- When the Content item ceases to be bound to an AD. In this case the ADID shall be set to zero.

8.2.2.7 Descrambling key

This field may be updated by Processing or Storage Entities.

The new value of CPCM Content descrambling_key shall be set randomly.

The C&R regime will set a maximum time duration during which a single descrambling key may be used. After that time, a new descrambling key shall be generated randomly and correspondingly a new Content Licence shall be issued.

8.2.2.8 Usage State Information

This field may only be updated by CPCM Instances that hosts an Acquisition Point or a Storage Entity.

8.2.2.9 AD Secret Signature

This field is updated if the Content Licence is under the direct protection of the AD Secret and if any other field has been changed in the Content Licence.

It is computed as the Message Authentication Code (see TS 102 825-5 [i.8]) of all other fields of the CPCM Content Licence.

Computation of the AD Secret signature is part of the Content Licence protection process (see clause 8.4).

8.3 CPCM Content Licence identification

8.3.1 General

The CPCM System manages CPCM Content Licences independently of the Content items to which they apply. This gives the benefit of CPCM more easily being able to interoperate across several home network ecosystems, 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).

As specified in clause 5.3.4, the CLID consists of an allocation scheme code and an identifier that is unique within each allocation scheme.

The 120 bits of scheme-specific Content Licence identifier is divided into a pre-allocated part and a part freely allocate-able by the Acquiring CPCM Instance. The respective sizes of these two parts vary according to the allocation scheme.

The extended generic structure (compared to the generic structure specified in clause 5.3.4) of the CPCM Content Licence identifier is shown in figure 43.

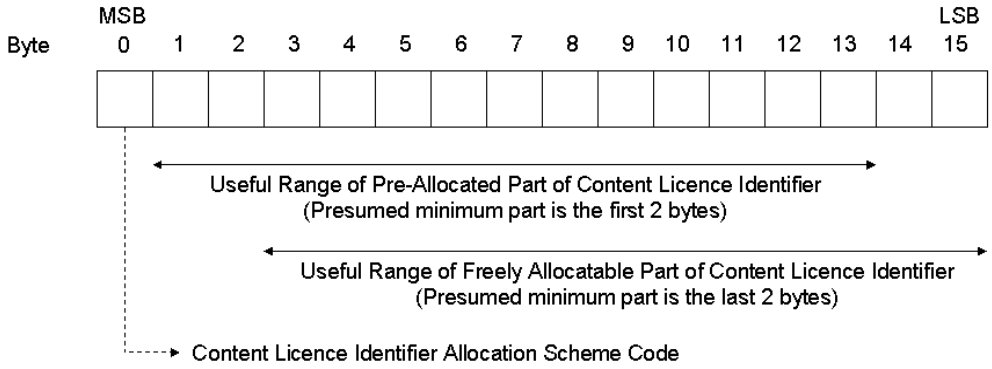

Figure 43: CLID extended generic structure

Table 238 shows the list of CLID allocation schemes provided by the present CPCM specification, and how the scheme-specific Content Licence identifier is structured.

A new CLID shall be allocated whenever a CL is generated for a new CPCM Content item that is Acquired.

The Acquisition Point that Acquires a CPCM Content item shall apply one of these CLID allocation schemes when generating the CLID for that Content item.

A new CL may be generated upon either a Storage or Processing operation that occurs on an existing CPCM Content item. In these cases the CPCM Functional Entity shall adapt the CLID of the original Content item CL so that the original allocation scheme is perpetuated, but this may necessitate the application of its own CIC Identifier as an element of the new CLID, depending on the allocation scheme applied.

Table 238: CPCM CLID allocation schemes

Allocation scheme code	Allocation scheme	Pre-allocated part length (Bytes)	Freely allocate-able part length (Bytes)
0x00	Reserved	Not applicable	Not applicable
0x01	DVB Broadcast Event and Acquisition Device	14	1
0x02	DVB Broadcast Service and Acquisition Device	12	3
0x03	CA System and Acquisition Device	10	5
0x04	DVB-MHP Application Identifier and Acquisition Device	14	1
0x05	Acquisition Device (only)	8	7
0x06	ISAN	12	3
0x07	Truncated ISAN	8	7
0x08 - 0xFF	Reserved for future use	Not applicable	Not applicable

Each CLID allocation scheme specifies its own method to assign unique Content Licence identifiers within the respective scheme. These are specified in the following sub-sections.

8.3.2 DVB Broadcast Event and Acquisition Device

In this scheme the pre-allocated part of the scheme-specific Content Licence identifier consists of the DVB broadcast event triplet concatenated with the CIC Identifier of the Acquiring CPCM Instance, and is thus 14 bytes in length. The freely allocatable part is the remaining one byte.

This CLID allocation scheme is depicted in figure 44.

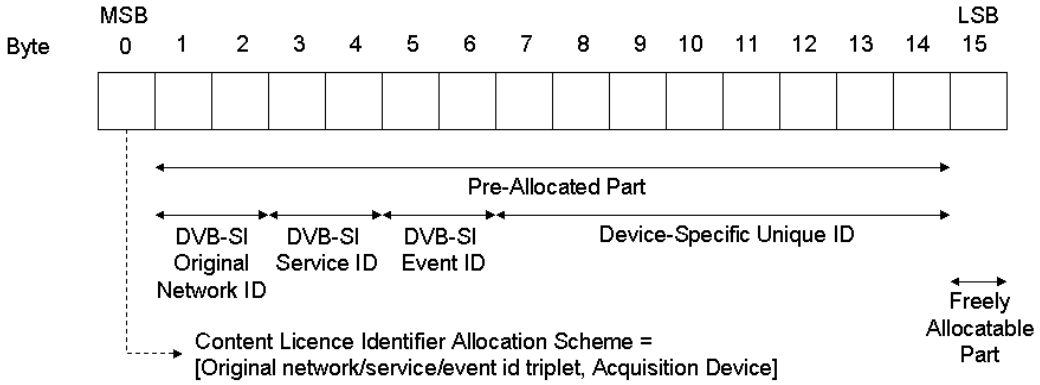

Figure 44: CLID allocation scheme for DVB broadcast event and Acquisition Device

8.3.3 DVB broadcast service and Acquisition Device

In this scheme the pre-allocated part of the scheme-specific Content Licence identifier consists of the DVB broadcast original_network_id plus service_id concatenated with the CIC Identifier of the Acquiring CPCM Instance, and is thus 12 bytes in length. The freely allocatable part is the remaining 3 bytes.

This CLID allocation scheme is depicted in figure 45.

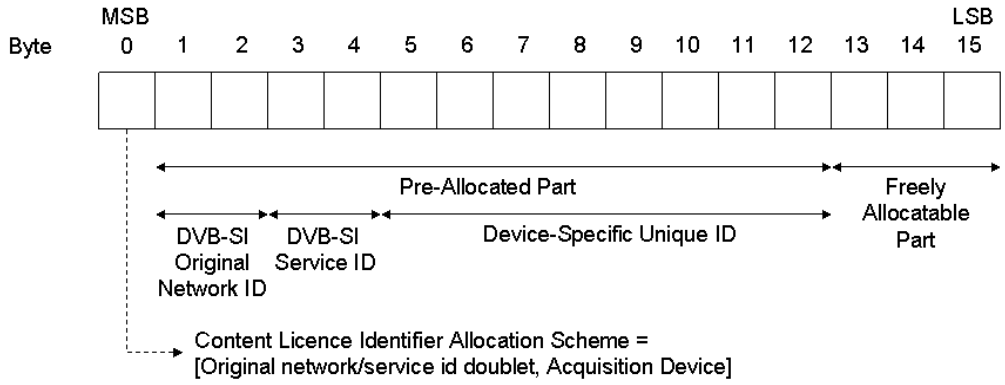

Figure 45: CLID allocation scheme for DVB broadcast service and Acquisition Device

8.3.4 CA system and Acquisition Device

In this scheme the pre-allocated part of the scheme-specific Content Licence identifier consists of the DVB broadcast CA system identifier of the CA system delivering the Input Content concatenated with the CIC Identifier of the Acquiring CPCM Instance, and is thus 10 bytes in length. The freely allocatable part is the remaining 5 bytes.

This CLID allocation scheme is depicted in figure 46.

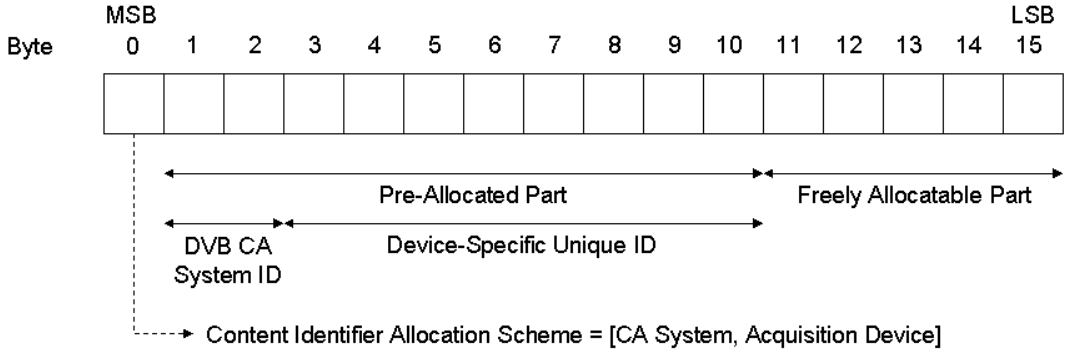

Figure 46: CLID allocation scheme for CA system and Acquisition Device

8.3.5 DVB-MHP application identifier and Acquisition Device

In this scheme the pre-allocated part of the scheme-specific Content Licence identifier consists of the MHP application identifier of the MHP application running in the CPCM Device hosting the Acquisition Point receiving the Input Content, concatenated with the CIC Identifier of the Acquiring CPCM Instance, and is thus 14 bytes in length. The freely allocatable part is the remaining 1 byte.

This CLID allocation scheme is depicted in figure 47.

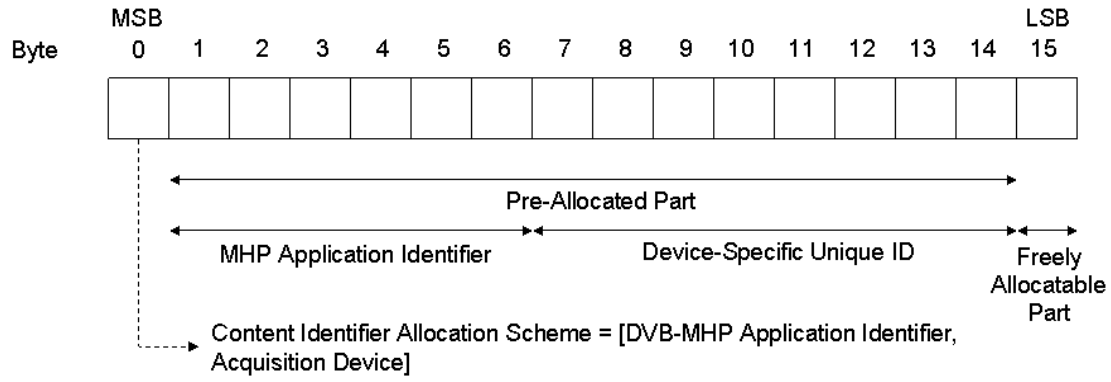

Figure 47: CLID allocation scheme for MHP application identifier and Acquisition Device

8.3.6 Acquisition Device (only)

In this scheme the pre-allocated part of the scheme-specific Content Licence identifier consists of the CIC Identifier of the Acquiring CPCM Instance, and is thus 8 bytes in length. The freely allocatable part is the remaining 7 bytes.

This CLID allocation scheme is depicted in figure 48.

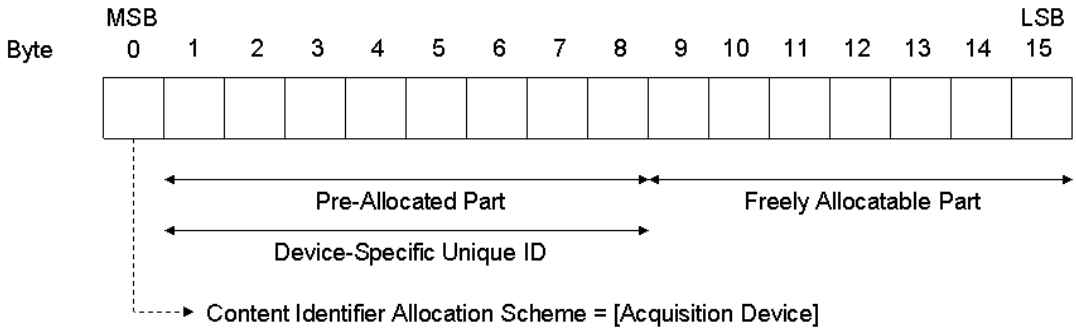

Figure 48: CLID allocation scheme for Acquisition Device (only)

8.3.7 ISAN

In this scheme the pre-allocated part of the scheme-specific Content Licence identifier consists of the ISAN of the Input Content item, and is thus 12 bytes in length. The freely allocatable part is the remaining 3 bytes.

This CLID allocation scheme is depicted in fFigure 49.

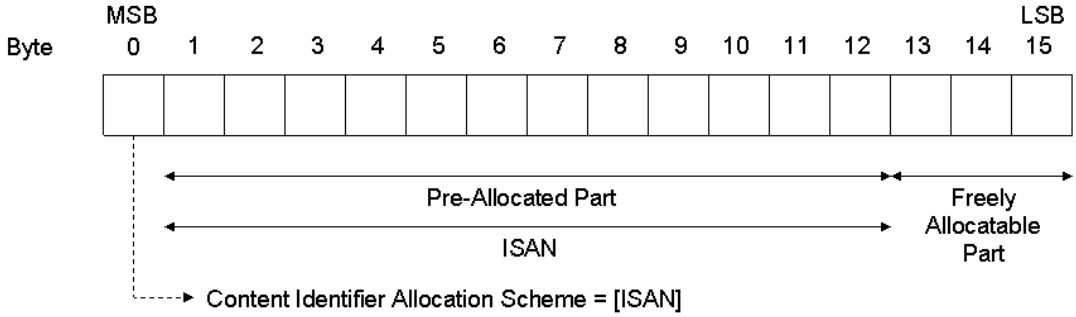

Figure 49: CLID allocation scheme for ISAN

8.3.8 Truncated ISAN

In this scheme the pre-allocated part of the scheme-specific Content Licence identifier consists of the truncated ISAN of the Input Content item, and is thus 8 bytes in length. The freely allocatable part is the remaining 7 bytes.

This CLID allocation scheme is depicted in figure 50.

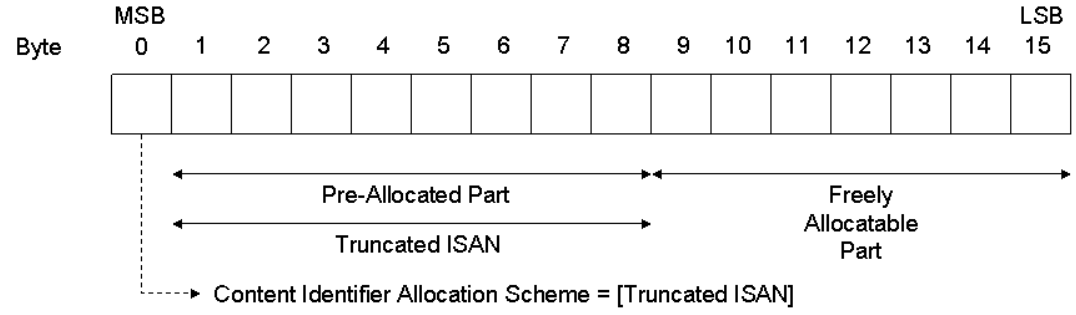

Figure 50: CLID allocation scheme for truncated ISAN

8.4 CPCM Content Licence verification

When receiving a Content Licence, a CPCM Instance shall perform the following process:

- Verify that its certificate has not expired;
- Verify that the right protection has been applied to the CL;
- Verify that the Content Licence signature is correct and, if applicable (i.e. if Content is not marked DNCS), decrypt the scrambling key using AD Secret or SAC session keys;
- Verify that it knows the Content Licence version;
- Verify that it implements at least one of the C&R regimes authorized in the Content Licence (see also clause 7.8);
- Verify that the authorized_domain_identifier matches either its ADID (if any) or is set to zero;
- Check that the CRL required for the valid C&R regime is available, and check that it is not revoked for this Content Item. If the Content Item is bound to an AD, then it shall also check that the corresponding AD has not been revoked.

It shall then proceed to USI verification and enforcement.

If any of the preceding conditions or steps in this clause fails, the CPCM Instance shall not handle the Content.

8.5 Content Licence re-Acquisition

The CPCM System includes a mechanism whereby a CPCM Instance that has already obtained a CL for a Content item is able to request the CPCM Instance that Acquired that Content item to provide a new CL, for example to obtain extended Authorized Usage for the Content item from the appropriate AAA. A basic framework for initiating the request and for the delivery of the new CL is defined, but all aspects of the transaction of obtaining extended rights are outside the scope of the CPCM System, and are implemented in conjunction with the Acquisition Point.

The mechanism for CL re-acquisition assumes that the CPCM Device that performed the original Acquisition also acts as the gateway to the service or facility that provides the extended rights. The Content Licence Creator (CLC) field in the CL is the CPCM Instance Certificate identifier of the CPCM Instance that originally Acquired the Content item, so it is this CPCM Instance that would also be able to provide extended Authorized Usage, and hence a new CL for that Content item.

Figure 51 shows the sequence of CPCM protocol calls used in order to re-acquire a CPCM CL, whereby CPCM Instance B already has a CL for the CPCM Content item in question, but the currently associated USI does not enable the desired Authorized Usage. A CPCM_get_new_CL call is thus issued to CPCM Instance A, being indicated as the CLC for the Content item in question. CPCM Instance A obtains the external service-specific rights for the Content item and then uses the CPCM_put_CL protocol call to provided the new version of the CL for the same Content item to CPCM Instance B, which shall discard the old CL for that Content item.

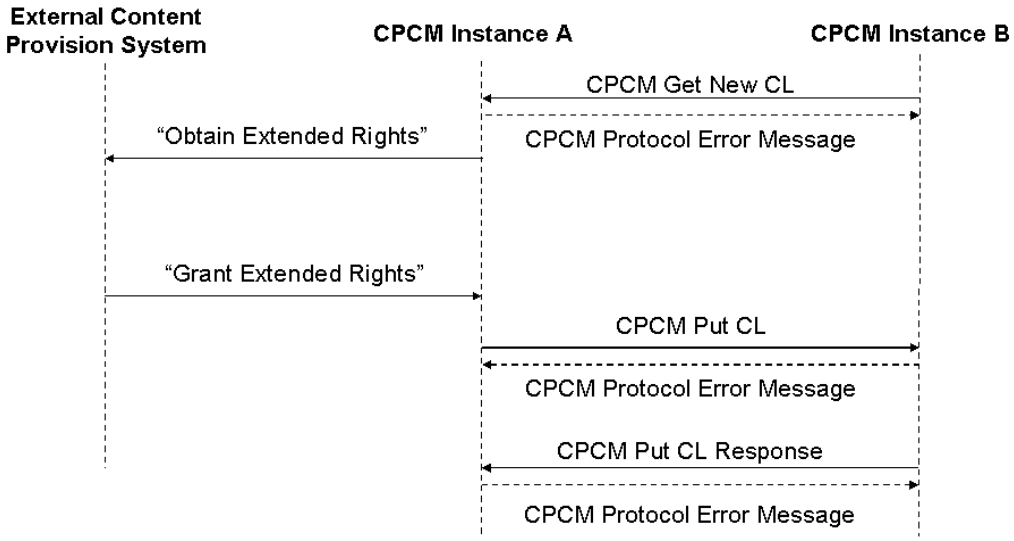

Figure 51: Sequence diagram for re-Acquisition of the CL

This mechanism is used in some example CPCM Content usage scenarios described in TR 102 825-11 [i.7].

In order to acquire a new CL for a CPCM Content item, a CPCM Instance shall discover whether or not the CPCM Instance that originally created the CL (i.e. the CPCM Instance whose CIC Identifier matches the CLC) is present. If not, the CPCM Instance may not re-acquire a new Content Licence, else it shall send to the latter CPCM Instance a *get_new_CL_message* carrying the current Content Licence and, optionally, the requested USI.

An Acquisition Point receiving *CPCM_get_new_CL_message* shall first verify the Content Licence and check that the CLC matches its CIC Identifier. Else, it denies the request. Then, it shall check with its CA/DRM or with an AAA whether it can deliver a new CL as requested by the user. New CL. An Acquisition Point that has no CA/DRM and that cannot contact a relevant AAA may not deliver a new CL. If the CA/DRM or the AAA accepts the request, it generates a new CL (as described in clause 8.2.1) and sends it to the requesting CPCM Instance using *CPCM_put_CL_message*.

8.6 Content Licence protection

8.6.1 CL protection modes

CPCM Content is always bound to some entity, either to an AD, or to the Source CPCM Device, or to the Destination CPCM Device, or to the SAC session between the Source and Destination CPCM Devices.

The Content Licence can require one of the following three types of protection, which in turn protects the USI contained within the Content Licence:

- SAC key (during transport) or device key (during storage) protected. In this mode, protection with the device key is governed by the C&R regime. Protection with the SAC key is done through the signing and the encryption of the message transmitting the Content Licence.
- The Content Licence may be sent separately from the Content, either by using the CPCM Content Licence exchange protocols, or the Content Licence exchange protocol call may be embedded in the Content item.
- AD Secret protection. This can be implemented using one of the two following methods:
  - Direct protection with the AD Secret (whereby the Content Licence shall be signed using the AD secret signature, then the descrambling key shall be encrypted using the AD Secret). The Content Licence needs no additional protection and can be sent in the clear or embedded in the content. This mode can be used for both Content Licence storage and protection. This mode enables the bit bucket scenario among CPCM Devices that are members of the same AD.

- Protection with the device key during storage and SAC key during transport. Before sending the Content Licence to a device from the same AD, the CPCM Instance needs first to ensure that the receiving CPCM Instance is a member of the same AD. This is achieved by applying the CPCM AD membership challenge protocol.

  In case of success, the C&R regime will define under which conditions a renewed testing may be omitted.

  AD secret protection shall be used only for exchanging content within a same AD and only for AD aware devices.

  CPCM Instances that are AD aware shall implement both methods. The choice of protection method for a particular Content Licence is left to the implementer (see TR 102 825-12 [i.12] for guidance).

Whatever the method used is, the AD shall not be revoked to use this kind of protection.

- AD secret and SAC key protection. These can be implemented using two ways:

  - Protections can be cumulated: content is first directly protected with the AD secret (i.e. signed and partially encrypted with it; cf. above). The AD protected CL is then to be signed and encrypted with the device secret (during storage) or the SAC secret (during transport).

  - Protection with the device key during storage and SAC key during transport. Before sending the CL, the CPCM Instance needs first to ensure that the receiving CPCM instance is member of the same Authorized Domain. To that end, it shall send a challenge in message AD_membership_challenge(). The receiving CPCM instance shall answer with a signature (using AD secret) of the message using message AD_membership_response(). In case of success, C&R regime will define under which conditions a new test may be skipped.

    Each CPCM Instance shall implement both methods. Choice of the method for a particular Content Licence is left to the implementer (see TR 102 825-12 [i.12] for guidance).

  Whatever the method used is, the AD shall not be revoked to use this kind of protection.

A Content Licence protected by the SAC session key (with or without AD Secret protection) may not be delivered twice using the same SAC session key. The Source CPCM Instance shall first renew the SAC. If a Destination CPCM Instance receives the same Content Licence protected by the same key twice, it shall renew the SAC and request the Content Licence again.

If the Content Licence is transferred through the SAC, the Source CPCM Instance shall first verify that the Sink CPCM Instance Certificate has not been revoked for the associated CPCM Content item.

8.6.2 CL protection mode usage

Content Licence protection modes shall be applied in accordance with the following rules:

- AD secret **and** SAC key protection shall be applied if content propagation occurs within the same AD and at least one other USI (i.e. USI that is not a Propagation control USI) requires SAC key protection;

- AD secret protection shall be applied if content propagation occurs within the same AD and no other USI (i.e. USI that is not Propagation control USI) requires SAC key protection;

- SAC key protection shall be applied if at least one USI requires SAC key protection and content propagation occurs regardless of AD binding;

- AD secret **or** SAC key protection shall be applied if content propagation occurs outside the AD and no USI requires SAC key protection.

If the Content is marked MLocal or VLocal (but not MCPCM or VCPCM, which are broader than MLocal or VLocal), Propagation is permitted within the same AD or within the Local Environment. CL shall be protected accordingly (i.e. AD secret protection for transfers within the same AD or SAC/Device key protection for transfers within the Local Environment) but does not need to be doubly protected. In other words, Content marked CCNA, Viewable, MAD/VAD and MLocal/VLocal with no further construction requires a CL protected with AD Secret when propagated within the AD or with SAC key when propagated outside the AD but does not require both protection.

Table 239 shows which USI requires which CL protection mode.

Table 239: CL required protection level

USI field name	Protection Mode
CCNA	-
C1	SAC/Device
CNM	SAC/Device
CN	SAC/Device
Zero Retention	SAC/Device
Viewable (V)	-
View Window Activated (VWA)	-
View Period Activated (VPA)	SAC/Device
Simultaneous View Count Activated (SVCA)	-
MLAD	AD
MGAD	AD
MAD	AD
MCPCM	-
VLAD	AD
VGAD	AD
VAD	AD
VCPCM	-
Remote Access Rule	-
MLocal	SAC/Device
VLocal	SAC/Device
Export/Output Controlled CPS	-
Export Beyond Trust	-
Disable Analogue SD Export	-
Disable Analogue SD Consumption	-
Disable Analogue HD Export	-
Disable Analogue HD Consumption	-
Image Constraint	-
Do Not CPCM Scramble	-

NOTE: AD Secret protection requires that the CPCM Instances dealing with the Content are AD aware and members of the same Authorized Domain. It enables storage on a bit bucket device.

8.6.3 CL protection mode changes

When Content is Copied or Moved (under the control of the USI) from one Source Instance to another Destination Instance, the protection mode of the Content Licence may vary as follows:

- If Content is Moved between CPCM Instances that are members of the same AD, then the protection method shall not change.

- If Content is Moved between CPCM Instances that are members of different ADs, or between an AD member and an Instance that is not a member of any AD, or between two CPCM Instances that are not members of any AD (NOTE: this Movement is allowed only if Content is marked MCPCM, VCPCM, MLocal or VLocal), then the Source Instance removes original Content Licence protection (that may be AD protection or SAC protection), sets authorized_domain_identifier to all zeros and protects the Content Licence using the SAC key. The Destination Instance verifies and removes SAC protection and sets the relevant protection:

  - If it is AD member, it sets authorized_domain_identifier to the identifier of its own AD and protects the Content Licence using AD protection.

  - Else, it protects the Content Licence using its Device key.

The above cases apply also when content is Moved for the purpose of Consumption or Export. However, the final CL protection shall be performed only if the Content is Stored.

List of figures

List of tables

History

Document history		
V1.1.1	July 2008	Publication